CECS 323 Term Project

TuTh 2 PM

Ruben Baerga, Scott Roberts, Francisco Fierro and Jesus Vargas

Esteemed Professionals in Database Whispering

**Classes and Associations**

A Customer owns one to many Vehicles.

A Vehicle is owned by one and only one Customer.

A Member is a Customer

A SteadyCustomer is a Member

A PremierCustomer is a Member

A ProspectiveCustomer is a Customer

A SteadyCustomerEmail is an Email

A ProspectiveCustomerEmail is an Email

A NotificationEmail is a SteadyCustomerEmail

A ReplyEmail is a SteadyCustomerEmail

A ConfirmationEmail is a SteadyCustomerEmail

A Vehicle is referenced in one to many SteadyCustomerEmails.

A SteadyCustomerEmail references one and only Vehicle.

A SteadyCustomerVehicle is a Vehicle

A PremierCustomerVehicle is a Vehicle

A SteadyCustomer owns one to many SteadyCustomerVehicles

A SteadyCustomerVehicle is owned by one and only one SteadyCustomer

A PremierCustomer owns one to many PremierCustomerVehicles

A PremierCustomerVehicle is owned by one and only one PremierCustomer

A Vehicle is serviced by one and only one Service Technician.

A Service Technician services one to many Vehicles.

A ServiceTechnician writes one to many Invoices.

An Invoice is written by one and only one ServiceTechnician.

An Invoice contains one to many InvoiceItems.

An Invoice Item is contained in one and only one Invoice.

An InvoiceItem is completed by one and only one Mechanic.

A Mechanic completes one to many InvoiceItems.

A Mechanic mentors zero to many Mentorships

A Mentorship is mentored by one and only one Mechanic

A Mentorship mentors one and only one Mechanic

A Mechanic is mentored in zero to many Mentorships

A MaintenaceItem exists on one to many InvoiceItems

An InvoiceItem exists as one and only one InvoiceItem

A MaintenancePackage is an aggregate of one to many MaintenanceItems

A Vehicle requires one to many MaintenancePackages.

A MaintenancePackage is required by one and only one Vehicle.

A SteadyCustomer is addressed in one to many SteadyCustomerEmails.

A SteadyCustomerEmail has an address from one and only one SteadyCustomer.

A SteadyCustomers refers one to many ProspectiveCustomers.

A ProspectiveCustomer is referred by one and only one SteadyCustomer.

A SteadyCustomer owns one to many SteadyCustomerVehicles.

A SteadyCustomerVehcile is owned by one and only one SteadyCustomer.

A PremierCustomer owns one to many PremierCustomerVehicles.

A PremierCustomerVehicle is owned by one and only one PremierCustomer.

A ProspectiveCustomer receives one to many ProspectiveCustomerEmails.

A ProspectiveCustomerEmail is received by one and only one ProspectiveCustomer.

A CorporateCustomer eats one to many Donuts.

A Donut is eaten by one and only one CorporateCustomer.

A CorporateCustomer drinks one to many Coffees.

A Coffee is drunk by one and only one CorporateCustomer.

**DDL Table Creation**

--Vehicles

CREATE TABLE vehicles (

vehicleID int NOT NULL,

customerID int NOT NULL,

mileage int NOT NULL,

make varchar(50) NOT NULL,

model varchar(50) NOT NULL,

CONSTRAINT vehicles\_pk PRIMARY KEY (vehicleID),

CONSTRAINT vehicles\_fk1 FOREIGN KEY (customerID) REFERENCES customers(customerID)

);

CREATE TABLE steadyCustomerVehicles (

vehicleID int NOT NULL,

steadyID int NOT NULL,

mileageEstimate int NOT NULL,

CONSTRAINT steadyCustomerVehicles\_pk PRIMARY KEY (vehicleID),

CONSTRAINT steadyCustomerVehicles\_fk1 FOREIGN KEY (vehicleID) REFERENCES vehicles (vehicleID),

CONSTRAINT steadyCustomerVehicles\_fk2 FOREIGN KEY (steadyID) REFERENCES steadyCustomers (steadyID)

);

CREATE TABLE mileagePredictions(

mileagePredictionID int NOT NULL,

vehicleID int NOT NULL,

mileagePrediction int NOT NULL,

mileageActual int,

"year" int NOT NULL,

CONSTRAINT mileagePredictions\_pk PRIMARY KEY (mileagePredictionID),

CONSTRAINT mileagePredictions\_fk FOREIGN KEY (vehicleID) REFERENCES premierCustomerVehicles (vehicleID)

);

CREATE TABLE premierCustomerVehicles (

vehicleID int NOT NULL,

premierID int NOT NULL,

servicableVehicleFlag boolean,

CONSTRAINT premierCustomerVehicles\_pk PRIMARY KEY (vehicleID),

CONSTRAINT premierCustomerVehicles\_fk1 FOREIGN KEY (vehicleID) REFERENCES vehicles(vehicleID),

CONSTRAINT premierCustomerVehicles\_fk2 FOREIGN KEY (premierID) REFERENCES premierCustomers(premierID)

);

--Mechanics

CREATE TABLE mechanics (

mechanicID int NOT NULL,

name varchar(50) NOT NULL,

DOB date NOT NULL,

CONSTRAINT mechanics\_pk PRIMARY KEY (mechanicID)

);

CREATE TABLE skills(

skillName varchar(50) NOT NULL,

CONSTRAINT skills\_pk PRIMARY KEY (skillName)

);

CREATE TABLE seniorMechanics (

mechanicID int NOT NULL,

salaryBonus int NOT NULL,

CONSTRAINT seniorMechanics\_pk PRIMARY KEY (mechanicID),

CONSTRAINT seniorMechanics\_fk FOREIGN KEY (mechanicID) REFERENCES mechanics(mechanicID)

);

CREATE TABLE skillsList(

mechanicID int NOT NULL,

skillName varchar(50) NOT NULL,

CONSTRAINT skillsList\_pk PRIMARY KEY (mechanicID, skillName),

CONSTRAINT skillsList\_fk1 FOREIGN KEY (mechanicID) REFERENCES mechanics(mechanicID),

CONSTRAINT skillsList\_fk2 FOREIGN KEY (skillName) REFERENCES skills(skillName)

);

CREATE TABLE mentorships (

mechanicID int NOT NULL,

mentorID int NOT NULL,

skillName VARCHAR(50) NOT NULL,

startDate date NOT NULL,

endDate date,

CONSTRAINT mentorships\_pk PRIMARY KEY (mechanicID, mentorID, skillName, startDate),

CONSTRAINT mentorships\_fk1 FOREIGN KEY (mechanicID) REFERENCES mechanics(mechanicID),

CONSTRAINT mentorships\_fk2 FOREIGN KEY (mentorID) REFERENCES mechanics(mechanicID),

CONSTRAINT mentorships\_fk3 FOREIGN KEY (skillName) REFERENCES skills(skillName)

);

--Maintance

CREATE TABLE serviceTechnicians (

serviceTechID int NOT NULL,

name varchar(50),

DOB DATE,

licenseNumber int,

CONSTRAINT serviceTechnicians\_pk PRIMARY KEY (serviceTechID)

);

--Check totalCost and see if it applies?

CREATE TABLE invoices (

invoiceID int NOT NULL,

vehicleID int NOT NULL,

serviceTechID int NOT NULL,

cDate date NOT NULL, --date of purchase

CONSTRAINT invoices\_pk PRIMARY KEY (invoiceID),

CONSTRAINT invoices\_fk1 FOREIGN KEY (vehicleID) REFERENCES vehicles(vehicleID),

CONSTRAINT invoices\_fk2 FOREIGN KEY (serviceTechID) REFERENCES serviceTechnicians(serviceTechID)

);

CREATE TABLE invoiceItems (

invoiceID int NOT NULL,

itemName varchar(50) NOT NULL,

skillName varchar(50) NOT NULL,

mechanicID int NOT NULL,

CONSTRAINT invoiceItems\_pk PRIMARY KEY (invoiceID, itemName, skillName, mechanicID),

CONSTRAINT invoiceItems\_fk1 FOREIGN KEY (invoiceID) REFERENCES invoices(invoiceID),

CONSTRAINT invoiceItems\_fk2 FOREIGN KEY (itemName, skillName) REFERENCES maintenanceItems(itemName, skillName),

CONSTRAINT invoiceItems\_fk3 FOREIGN KEY (mechanicID) REFERENCES mechanics(mechanicID)

);

CREATE TABLE maintenancePackages (

packageID int NOT NULL,

vehicleID int NOT NULL,

targetMileage int NOT NULL,

CONSTRAINT maintenancePackages\_pk PRIMARY KEY (packageID),

CONSTRAINT maintenancePackages\_fk FOREIGN KEY (vehicleID) REFERENCES vehicles(vehicleID)

);

CREATE TABLE maintenanceItems (

packageID int NOT NULL,

itemName varchar(50) NOT NULL,

skillName varchar(50) NOT NULL,

cost float NOT NULL,

price float NOT NULL,

CONSTRAINT maintenanceItems\_pk PRIMARY KEY (itemName, skillName),

CONSTRAINT maintenanceItems\_fk1 FOREIGN KEY (packageID) REFERENCES maintenancePackages(packageID),

CONSTRAINT maintenanceItems\_fk2 FOREIGN KEY (skillName) REFERENCES skills(skillName)

);

--Email

CREATE TABLE emails (

emailID int NOT NULL,

sendAddress varchar(50) NOT NULL,

receiveAddress varchar(50) NOT NULL,

sendDate date NOT NULL,

sendTime varchar(50) NOT NULL,

subject varchar(50),

CONSTRAINT emails\_pk PRIMARY KEY (emailID)

);

CREATE TABLE steadyCustomerEmails(

emailID int NOT NULL,

steadyID int NOT NULL,

vehicleID int NOT NULL,

CONSTRAINT steadyCustomerEmails\_pk PRIMARY KEY (emailID, steadyID, vehicleID),

CONSTRAINT steadyCustomerEmails\_fk1 FOREIGN KEY (emailID) REFERENCES emails(emailID),

CONSTRAINT steadyCustomerEmails\_fk2 FOREIGN KEY (steadyID) REFERENCES steadyCustomers(steadyID),

CONSTRAINT steadyCustomerEmails\_fk3 FOREIGN KEY (vehicleID) REFERENCES vehicles(vehicleID)

);

CREATE TABLE notificationEmails(

emailID int NOT NULL,

steadyID int NOT NULL,

vehicleID int NOT NULL,

targetMileage int NOT NULL,

CONSTRAINT notificationEmails\_pk PRIMARY KEY (emailID, steadyID, vehicleID),

CONSTRAINT notificationEmails\_fk FOREIGN KEY (emailID, steadyID, vehicleID) REFERENCES steadyCustomerEmails (emailID, steadyID, vehicleID)

);

CREATE TABLE replyEmails(

emailID int NOT NULL,

steadyID int NOT NULL,

vehicleID int NOT NULL,

startDate date,

endDate date,

CONSTRAINT replyEmails\_pk PRIMARY KEY (emailID, steadyID, vehicleID),

CONSTRAINT replyEmails\_fk FOREIGN KEY (emailID, steadyID, vehicleID) REFERENCES steadyCustomerEmails(emailID, steadyID, vehicleID)

);

CREATE TABLE confirmationEmails(

emailID int NOT NULL,

steadyID int NOT NULL,

vehicleID int NOT NULL,

appointmentDate date,

appointmentTime varchar(50),

CONSTRAINT confirmationEmails\_pk PRIMARY KEY (emailID, steadyID, vehicleID),

CONSTRAINT confirmationEmails\_fk FOREIGN KEY (emailID, steadyID, vehicleID) REFERENCES steadyCustomerEmails (emailID, steadyID, vehicleID)

);

CREATE TABLE prospectiveCustomerEmails(

emailID int NOT NULL,

prospectiveID int NOT NULL,

specialDealName varchar(50),

CONSTRAINT prospectiveCustomerEmails\_pk PRIMARY KEY (emailID, prospectiveID, specialDealName),

CONSTRAINT prospectiveCustomerEmails\_fk1 FOREIGN KEY (emailID) REFERENCES emails (emailID),

CONSTRAINT prospectiveCustomerEmails\_fk2 FOREIGN KEY (prospectiveID) REFERENCES prospectiveCustomers (prospectiveID)

);

--Customers

CREATE TABLE customers(

customerID int NOT NULL,

name varchar(50) NOT NULL,

DOB date NOT NULL,

emailAddress varchar(50) NOT NULL,

loyaltyPoints int,

yearsLoyal int,

CONSTRAINT customers\_pk PRIMARY KEY (customerID)

);

CREATE TABLE pets(

customerID int NOT NULL,

petID int NOT NULL,

petName varchar(50) NOT NULL,

petDOB date NOT NULL,

petSpecies varchar(50) NOT NULL,

CONSTRAINT pets\_pk PRIMARY KEY (customerID, petID),

CONSTRAINT pets\_fk FOREIGN KEY (customerID) REFERENCES customers(customerID)

);

CREATE TABLE zipLocations(

zipCode int NOT NULL,

city varchar(50) NOT NULL,

state varchar(50) NOT NULL,

CONSTRAINT zipLocations\_pk PRIMARY KEY (zipCode)

);

--Needs to be changed to Addresses

--Fix this up and figure out the connections

CREATE TABLE corporateaddresses(

corporateID int NOT NULL,

addressType varchar(50) NOT NULL,

address varchar(50) NOT NULL,

zipCode int NOT NULL,

CONSTRAINT corporateAddresses\_pk PRIMARY KEY (address, corporateID),

CONSTRAINT corporateAddresses\_fk1 FOREIGN KEY (zipCode) REFERENCES zipLocations(zipCode),

CONSTRAINt corporateAddresses\_fk2 FOREIGN KEY (corporateID) REFERENCES corporateCustomers(corporateID)

);

CREATE TABLE privateaddresses(

privateID int NOT NULL,

address varchar(50) NOT NULL,

zipCode int NOT NULL,

CONSTRAINT privateAddresses\_pk PRIMARY KEY (address, privateID),

CONSTRAINT privateAddresses\_fk1 FOREIGN KEY (zipCode) REFERENCES zipLocations(zipCode),

CONSTRAINt privateAddresses\_fk2 FOREIGN KEY (privateID) REFERENCES privateCustomers(privateID)

);

CREATE TABLE privateCustomers(

privateID int NOT NULL,

CONSTRAINT privateCustomers\_pk PRIMARY KEY (privateID),

CONSTRAINT privateCustomers\_fk1 FOREIGN KEY (privateID) REFERENCES customers(customerID)

);

CREATE TABLE corporateCustomers(

corporateID int NOT NULL,

CONSTRAINT corporateCustomers\_pk PRIMARY KEY (corporateID),

CONSTRAINT corporateCustomers\_fk1 FOREIGN KEY (corporateID) REFERENCES customers(customerID)

);

CREATE TABLE donuts(

corporateID int NOT NULL,

dateEaten date NOT NULL,

timeEaten varchar(50) NOT NULL,

CONSTRAINT donuts\_pk PRIMARY KEY (corporateID, dateEaten, timeEaten),

CONSTRAINT donuts\_fk FOREIGN KEY (corporateID) REFERENCES corporateCustomers(corporateID)

);

CREATE TABLE referrals(

referralID int NOT NULL,

name varchar(50) NOT NULL,

DOB date NOT NULL,

emailAddress varchar(50) NOT NULL,

CONSTRAINT referral\_pk PRIMARY KEY (referralID)

);

CREATE TABLE members(

memberID int NOT NULL,

referralID int NOT NULL,

CONSTRAINT members\_pk PRIMARY KEY (memberID),

CONSTRAINT members\_fk1 FOREIGN KEY (memberID) REFERENCES customers(customerID),

CONSTRAINT members\_fk2 FOREIGN KEY (referralID) REFERENCES referrals(referralID)

);

CREATE TABLE payments(

memberID int NOT NULL,

amount float NOT NULL,

payDate date NOT NULL,

CONSTRAINT payments\_pk PRIMARY KEY (memberID, amount, payDate),

CONSTRAINT payments\_fk FOREIGN KEY (memberID) REFERENCES members(memberID)

);

CREATE TABLE prospectiveCustomers(

prospectiveID int NOT NULL,

emailCount int NOT NULL,

deadProspectFlag boolean NOT NULL,

CONSTRAINT prospectiveCustomers\_pk PRIMARY KEY (prospectiveID),

CONSTRAINT prospectiveCustomers\_fk FOREIGN KEY (prospectiveID) REFERENCES customers(customerID)

);

CREATE TABLE premierCustomers(

premierID int NOT NULL,

CONSTRAINT premierCustomers\_pk PRIMARY KEY (premierID),

CONSTRAINT premierCustomers\_fk FOREIGN KEY (premierID) REFERENCES members(memberID)

);

CREATE TABLE annualFees(

premierID int NOT NULL,

fee DECIMAL(20,2) NOT NULL,

"year" int NOT NULL,

CONSTRAINT annualFees\_pk PRIMARY KEY (premierID, "year"),

CONSTRAINT annualFees\_fk FOREIGN KEY (premierID) REFERENCES premierCustomers(premierID)

);

CREATE TABLE steadyCustomers(

steadyID int NOT NULL,

recentPayment int NOT NULL,

CONSTRAINT steadyCustomers\_pk PRIMARY KEY (steadyID),

CONSTRAINT steadyCustomers\_fk FOREIGN KEY (steadyID) REFERENCES members(memberID)

);

**DML Data Language**

insert into vehicles (vehicleID, customerID, mileage, make, model) values

(1, 1, 1000, 'Toyota', 'Camry'), --Steady/Corporate

(2, 1, 100001, 'Toyota', 'Corolla'),

(3, 2, 9892, 'Toyota', 'Wish'), --Steady/Private

(4, 2, 28382, 'Toyota', 'Yaris'),

(5, 3, 32833, 'Honda', 'CR-V'), --Premier/Corporate

(6, 4, 25433, 'Ford','Focus'), --Premier/Private

(7, 5, 8383, 'Hyundai','Tucson'), --Steady/Corporate

(8, 5, 9531, 'Infiniti','Q30'),

(9, 5, 7384, 'Kia','Soul'),

(10, 6, 18280, 'Mazda','Mazda2'), --Premier/Corporate

(11, 7, 12322, 'Volkswagen','Beetle'), --Prospective/Private

(12, 8, 50542, 'Volkswagen','Jetta'); --Prospective/Corporate

--mileageEstimate is an estimate number of miles they will drive,

--not the estimate mileage on the car when they bring it in.

insert into steadyCustomerVehicles (vehicleID, steadyID, mileageEstimate) values

(1, 1, 3000),

(2, 1, 2000),

(3, 2, 5000),

(4, 2, 200),

(7, 5, 500),

(8, 5, 600),

(9, 5, 700);

--For premier customers

insert into mileagePredictions (mileagePredictionID, vehicleID, mileagePrediction, mileageActual, "year") values

(1, 5, 10000, 9700, 2016),

(2, 5, 9700, null, 2017),

(3, 6, 15000, 20000, 2016),

(4, 6, 20000, null, 2017),

(5, 10, 30000, 10000, 2016),

(6, 10, 10000, null, 2017);

insert into premierCustomerVehicles (vehicleID, premierID, servicableVehicleFlag) values

(5, 3, true),

(6, 4, true),

(10, 6, true);

insert into mechanics (mechanicID, name, DOB) values

(100, 'Mac', '1980-01-12'),

(101, 'Jess', '1985-03-23'),

(102, 'Bill', '2000-10-11'),

(103, 'Mike', '1995-10-11'),

(104, 'Kelly', '1991-05-15');

insert into skills (skillName) values

('Change Oil'),

('Change Battery'),

('Remove Engine'),

('Replace Brake Pads'),

('Tune Engine'),

('Replace Tires'),

('Rotate Tires'),

('Replace Hoses'),

('Replace Transmission Fluid'),

('Fix Power Steering'),

('Change Air Filter'),

('Replace Windshield Wipers'),

('Add Muffler'),

('Add Tint'),

('Add Decal'),

('Add Sound System');

insert into seniorMechanics (mechanicID, salaryBonus) values

(101, 4250),

(103, 5000);

insert into skillsList (mechanicID, skillName) values

(100, 'Change Oil'), --Mac

(100, 'Replace Hoses'),

(100, 'Change Air Filter'),

(101, 'Change Oil'), --Jess

(101, 'Change Battery'),

(101, 'Remove Engine'),

(101, 'Replace Brake Pads'),

(101, 'Tune Engine'),

(101, 'Replace Tires'),

(101, 'Rotate Tires'),

(101, 'Replace Hoses'),

(101, 'Replace Transmission Fluid'),

(101, 'Fix Power Steering'),

(102, 'Add Tint'), --Bill

(102, 'Add Decal'),

(103, 'Add Sound System'), --Mike

(103, 'Add Decal'),

(103, 'Add Tint'),

(103, 'Add Muffler'),

(103, 'Replace Windshield Wipers'),

(103, 'Change Air Filter'),

(103, 'Fix Power Steering'),

(103, 'Replace Transmission Fluid'),

(103, 'Replace Hoses'),

(103, 'Rotate Tires'),

(103, 'Replace Tires'),

(103, 'Tune Engine'),

(103, 'Replace Brake Pads'),

(103, 'Remove Engine'),

(103, 'Change Battery'),

(103, 'Change Oil'),

(104, 'Replace Tires'), --Kelly

(104, 'Rotate Tires'),

(104, 'Replace Brake Pads'),

(104, 'Remove Engine'),

(104, 'Tune Engine');

insert into mentorships (mechanicID, mentorID, skillName, startDate, endDate) values

(102, 103, 'Add Tint', '2015-05-05', '2015-06-05'),

(100, 101, 'Replace Hoses', '2017-03-06', '2017-03-10'),

(100, 101, 'Replace Hoses', '2017-03-15', '2017-03-19'),

(104, 101, 'Replace Brake Pads', '2016-02-24', '2016-02-28'),

(104, 101, 'Change Battery', '2017-05-05', null),

(102, 103, 'Add Decal', '2015-06-06', '2015-07-01');

insert into serviceTechnicians (serviceTechID, name, DOB, licenseNumber) values

(100, 'Mac', '1980-01-12', 2316543),

(201, 'May', '1993-12-17', 7242847),

(202, 'Jack', '1994-07-14', 6634331),

(203, 'Scott', '1990-10-10', 2811215);

insert into invoices (invoiceID, vehicleID, serviceTechID, cDate) values

(1, 1, 100, '2016-04-04'),

(2, 2, 100, '2016-04-05'),

(3, 3, 201, '2016-04-06'),

(4, 4, 201, '2016-04-07'),

(5, 5, 201, '2016-04-08'),

(6, 6, 202, '2016-04-09'),

(7, 7, 202, '2016-04-10'),

(8, 8, 202, '2016-04-11'),

(9, 9, 203, '2016-04-12'),

(10, 10, 203, '2016-04-13');

--itemName connection to skills table?

insert into invoiceItems (invoiceID, itemName, skillName, mechanicID) values

(1, 'Change Oil', 'Change Oil', 103),

(1, 'Change Battery', 'Change Battery', 103),

(1, 'Rotate Tires', 'Rotate Tires', 103),

(2, 'Change Oil', 'Change Oil', 103),

(2, 'Change Air Filter', 'Change Air Filter', 103),

(2, 'Add Muffler', 'Add Muffler', 103),

(3, 'Change Oil', 'Change Oil', 103),

(3, 'Replace Windshield Wipers', 'Replace Windshield Wipers', 103),

(3, 'Replace Hoses', 'Replace Hoses', 103),

(4, 'Change Oil', 'Change Oil', 103),

(4, 'Add Sound System', 'Add Sound System', 103),

(4, 'Fix Power Steering', 'Fix Power Steering', 103),

(5, 'Tune Engine', 'Tune Engine', 103),

(5, 'Replace Hoses', 'Replace Hoses', 103),

(5, 'Change Oil', 'Change Oil', 103),

(6, 'Replace Windshield Wipers', 'Replace Windshield Wipers', 103),

(6, 'Replace Tires', 'Replace Tires', 103),

(6, 'Remove Engine', 'Remove Engine', 103),

(7, 'Change Oil', 'Change Oil', 103),

(7, 'Replace Tires', 'Replace Tires', 103),

(7, 'Change Air Filter', 'Change Air Filter', 103),

(8, 'Tune Engine', 'Tune Engine', 103),

(8, 'Replace Hoses', 'Replace Hoses', 103),

(8, 'Change Oil', 'Change Oil', 103),

(9, 'Change Oil', 'Change Oil', 103),

(9, 'Change Air Filter', 'Change Air Filter', 103),

(9, 'Change Battery', 'Change Battery', 103),

(10, 'Change Oil', 'Change Oil', 103),

(10, 'Replace Hoses', 'Replace Hoses', 103),

(10, 'Tune Engine', 'Tune Engine', 103);

insert into maintenancePackages (packageID, vehicleID, targetMileage) values

(1,1,5000),

(2,2,5000),

(3,3,2000),

(4,4,5000),

(5,5,4000),

(6,6,1000),

(7,7,10000),

(8,8,2000),

(9,9,5000),

(10,10,2000);

insert into maintenanceItems (packageID, itemName, skillName, cost, price) values

(1, 'Change Oil', 'Change Oil', 8.99, 30.00),

(1, 'Change Battery', 'Change Battery', 48.50, 66.45),

(1, 'Rotate Tires', 'Rotate Tires', 5.40, 13.23),

(2, 'Change Oil', 'Change Oil', 8.99, 30.00),

(2, 'Change Air Filter', 'Change Air Filter', 12.76, 15.22),

(2, 'Add Muffler', 'Add Muffler', 40.40, 62.22),

(3, 'Change Oil', 'Change Oil', 8.99, 30.00),

(3, 'Replace Windshield Wipers', 'Replace Windshield Wipers', 19.99, 30.21),

(3, 'Replace Hoses', 'Replace Hoses', 24.10, 34.34),

(4, 'Change Oil', 'Change Oil', 8.99, 30.00),

(4, 'Add Sound System', 'Add Sound System', 138.98, 160.33),

(4, 'Fix Power Steering', 'Fix Power Steering', 123.21, 150.00),

(5, 'Tune Engine', 'Tune Engine', 38.73, 72.23),

(5, 'Replace Hoses', 'Replace Hoses', 24.10, 34.34),

(5, 'Change Oil', 'Change Oil', 8.99, 30.00),

(6, 'Replace Windshield Wipers', 'Replace Windshield Wipers', 19.99, 23.33),

(6, 'Replace Tires', 'Replace Tires', 154.23, 183.23),

(6, 'Remove Engine', 'Remove Engine', 24.34, 43.58),

(7, 'Change Oil', 'Change Oil', 8.99, 30.00),

(7, 'Replace Tires', 'Replace Tires', 154.23, 183.33),

(7, 'Change Air Filter', 'Change Air Filter', 12.76, 15.44),

(8, 'Tune Engine', 'Tune Engine', 38.73, 54.34),

(8, 'Replace Hoses', 'Replace Hoses', 24.10, 34.23),

(8, 'Change Oil', 'Change Oil', 8.99, 30.00),

(9, 'Change Oil', 'Change Oil', 8.99, 30.00),

(9, 'Change Air Filter', 'Change Air Filter', 12.76, 15.43),

(9, 'Change Battery', 'Change Battery', 48.50, 66.66),

(10, 'Change Oil', 'Change Oil', 8.99, 30.00),

(10, 'Replace Hoses', 'Replace Hoses', 24.10, 34.43),

(10, 'Tune Engine', 'Tune Engine', 38.73, 65.45);

insert into emails (emailID, sendAddress, receiveAddress, sendDate, sendTime, subject) values

(1, 'dave@autoshop.com', 'frankthetank@csulb.edu', '2016-02-03', '12:30', 'This is your first contact'), --Steady 1

(2,'frankthetank@csulb.edu', 'dave@autoshop.com', '2016-02-04', '12:30', 'Reply back'),

(3, 'dave@autoshop.com', 'frankthetank@csulb.edu', '2016-02-05', '12:30', 'Actual Date'),

(4, 'dave@autoshop.com', 'frankthetank@csulb.edu', '2016-02-06', '12:30', 'This is your first contact'), --Steady 2

(5, 'frankthetank@csulb.edu', 'dave@autoshop.com', '2016-02-07', '12:30', 'Reply back'),

(6, 'dave@autoshop.com', 'frankthetank@csulb.edu', '2016-02-08', '12:30', 'Actual Date'),

(7, 'dave@autoshop.com', 'databasewhisperer@csulb.edu', '2016-02-09', '12:30', 'This is your first contact'), --Steady 3

(8,'databasewhisperer@csulb.edu', 'dave@autoshop.com', '2016-02-10', '12:30', 'Reply back'),

(9, 'dave@autoshop.com', 'databasewhisperer@csulb.edu', '2016-02-11', '12:30', 'Actual Date'),

(10, 'dave@autoshop.com', 'databasewhisperer@csulb.edu', '2016-02-12', '12:30', 'This is your first contact'),--Steady 4

(11,'databasewhisperer@csulb.edu', 'dave@autoshop.com', '2016-02-13', '12:30', 'Reply back'),

(12, 'dave@autoshop.com', 'databasewhisperer@csulb.edu', '2016-02-14', '12:30', 'Actual Date'),

(13, 'dave@autoshop.com', 'max@csulb.edu', '2016-02-15', '12:30', 'This is your first contact'), --Steady 5

(14, 'max@csulb.edu', 'dave@autoshop.com', '2016-02-16', '12:30', 'Reply back'),

(15, 'dave@autoshop.com', 'max@csulb.edu', '2016-02-17', '12:30', 'Actual Date'),

(16, 'dave@autoshop.com', 'max@csulb.edu', '2016-02-18', '12:30', 'This is your first contact'), --Steady 6

(17, 'max@csulb.edu', 'dave@autoshop.com', '2016-02-19', '12:30', 'Reply back'),

(18, 'dave@autoshop.com', 'max@csulb.edu', '2016-02-20', '12:30', 'Actual Date'),

(19, 'dave@autoshop.com', 'max@csulb.edu', '2016-02-21', '12:30', 'This is your first contact'), --Steady 7

(20, 'max@csulb.edu', 'dave@autoshop.com', '2016-02-22', '12:30', 'Reply back'),

(21, 'dave@autoshop.com', 'max@csulb.edu', '2016-02-23', '12:30', 'Actual Date'),

(22, 'dave@autoshop.com', 'grande@csulb.edu', '2016-02-24', '12:30', 'Special Deal'), --Prospective 1

(23, 'dave@autoshop.com', 'beatleslover@csulb.edu', '2016-02-25', '12:30', 'Special Deal'), --Prospective 2

(24, 'dave@autoshop.com', 'beatleslover@csulb.edu', '2016-02-26', '12:30', 'Special Deal'); --Prospective 3

insert into steadyCustomerEmails (emailID, steadyID, vehicleID) values

(1, 1, 1),

(2, 1, 1),

(3, 1, 1),

(4, 1, 2),

(5, 1, 2),

(6, 1, 2),

(7, 2, 3),

(8, 2, 3),

(9, 2, 3),

(10, 2, 4),

(11, 2, 4),

(12, 2, 4),

(13, 5, 7),

(14, 5, 7),

(15, 5, 7),

(16, 5, 8),

(17, 5, 8),

(18, 5, 8),

(19, 5, 9),

(20, 5, 9),

(21, 5, 9);

insert into notificationEmails (emailID, steadyID, vehicleID, targetMileage) values

(1, 1, 1, 5000),

(4, 1, 2, 5000),

(7, 2, 3, 2000),

(10, 2, 4, 5000),

(13, 5, 7, 10000),

(16, 5, 8, 2000),

(19, 5, 9, 5000);

insert into replyEmails (emailID, steadyID, vehicleID, startDate, endDate) values

(2, 1, 1, '2016-03-04', '2016-04-04'),

(5, 1, 2, '2016-03-05', '2016-04-05'),

(8, 2, 3, '2016-03-06', '2016-04-06'),

(11, 2, 4, '2016-03-07', '2016-04-07'),

(14, 5, 7, '2016-03-08', '2016-04-08'),

(17, 5, 8, '2016-03-09', '2016-04-09'),

(20, 5, 9, '2016-03-10', '2016-04-10');

insert into confirmationEmails (emailID, steadyID, vehicleID, appointmentDate, appointmentTime) values

(3, 1, 1, '2016-03-04', '1:00'),

(6, 1, 2, '2016-03-05', '2:00'),

(9, 2, 3, '2016-03-06', '11:00'),

(12, 2, 4, '2016-03-07', '8:00'),

(15, 5, 7, '2016-03-08', '4:00'),

(18, 5, 8, '2016-03-09', '2:00'),

(21, 5, 9, '2016-03-10', '1:00');

insert into prospectiveCustomerEmails (emailID, prospectiveID, specialDealName) values

(22, 7, 'Hooplah Sale!'),

(23, 8, 'Hooplah Sale!'),

(24, 8, 'Jump for Money!');

insert into pets (customerID, petID, petName, petDOB, petSpecies) values

(2, 666, 'Snuggles', '2013-02-01', 'Dog'),

(2, 667, 'Squanchy', '2013-02-02', 'Cat'),

(2, 668, 'Sasquach', '2013-02-03', 'Sasquach'),

(2, 669, 'Sam', '2013-02-04', 'Human');

insert into customers (customerID, name, DOB, emailAddress, loyaltyPoints, yearsLoyal) values

(1, 'Francisco', '1940-12-25', 'frankthetank@csulb.edu', 30, 2), --Steady/Corp.

(2, 'Ruben', '1984-03-02', 'databasewhisperer@csulb.edu', 40, 3), --Steady/Private

(3, 'Jesus', '2000-06-12', 'Jesusblesshisname@csulb.edu', 80, 5), --Premier/Corp.

(4, 'Scooter', '1990-01-01', 'scootermcgrooger@csulb.edu', 80, 5), --Premier/Private

(5, 'Max', '1960-11-21', 'max@csulb.edu', 160, 10), --Steady/Corp.

(6, 'Christian', '1994-12-25', 'sk8erpnk@csulb.edu', 50, 4), --Premier/Corp.

(7, 'Rio', '1999-07-12', 'grande@csulb.edu', null, null), --Prospective/Private

(8, 'Jude', '1995-11-10', 'beatleslover@csulb.edu', null, null); --Prospective/Corp.

insert into zipLocations (zipCode, city, state) values

(90630, 'Cypress', 'CA'),

(90840, 'Long Beach', 'CA'),

(90715, 'Long Beach', 'CA'),

(90812, 'Long Beach', 'CA'),

(90848, 'Long Beach', 'CA'),

(90720, 'Los Alamitos', 'CA'),

(90724, 'Los Alamitos', 'CA'),

(92625, 'Newport Beach', 'CA');

insert into corporateAddresses (corporateID, addressType, address, zipCode) values

(1, 'Home', '31231 Yada Yada', 90840), --Steady/Corp.

(3, 'Home', '32943 Blah Blah', 92625), --Premier/Corp.

(5, 'Home', '43445 Go Way', 90720), --Steady/Corp.

(6, 'Home', '47322 Iterative Stuff', 90630), --Premier/Corp.

(8, 'Home', '85847 My Home', 90840); --Prospective/Corp.

insert into privateAddresses (privateID, address, zipCode) values

(2, '28 Mulhollan Dr.', 90840), --Steady/Private

(4, '11346 Kent Way', 90630), --Premier/Private

(7, 'Rio Grande. River', 90724); --Prospective/Private

insert into privateCustomers (privateID) values

(2), --Steady/Private

(4), --Premier/Private

(7); --Prospective/Private

insert into corporateCustomers (corporateID) values

(1), --Steady/Corp.

(3), --Premier/Corp.

(5), --Steady/Corp.

(6), --Premier/Corp.

(8); --Prospective/Corp.

insert into donuts (corporateID, dateEaten, timeEaten) values

(1, '2016-04-04', '12:20'),

(3, '2016-04-05', '11:00'),

(5, '2016-04-06', '12:20'),

(5, '2016-03-07', '8:45'),

(5, '2016-03-08', '1:15'),

(6, '2016-03-09', '12:20'),

(8, '2016-03-10', '8:40');

insert into referrals (referralID, name, DOB, email) values

(1, 'Rio', '1999-07-12', 'grande@csulb.edu'),

(3, 'Jude', '1995-11-10', 'beatleslover@csulb.edu');

insert into members (memberID, referralID) values

(1, 1), --Steady/Corp.

(2, 1), --Steady/Private

(3, 3), --Premier/Corp.

(4, 3), --Premier/Private

(5, 1), --Steady/Corp.

(6, 1); --Premier/Corp.

insert into prospectiveCustomers (prospectiveID, emailCount, deadProspectFlag) values

(7, 1, false), --Prospective/Private

(8, 2, false); --Prospective/Corp.

insert into premierCustomers (premierID) values

(3), --Premier/Corp.

(4), --Premier/Private

(6); --Premier/Corp.

insert into annualFees (premierID, fee, "year") values

(3, 2125.43, 2016), --Premier/Corp.

(4, 2430.50, 2016), --Premier/Private

(6, 1895.25, 2016);

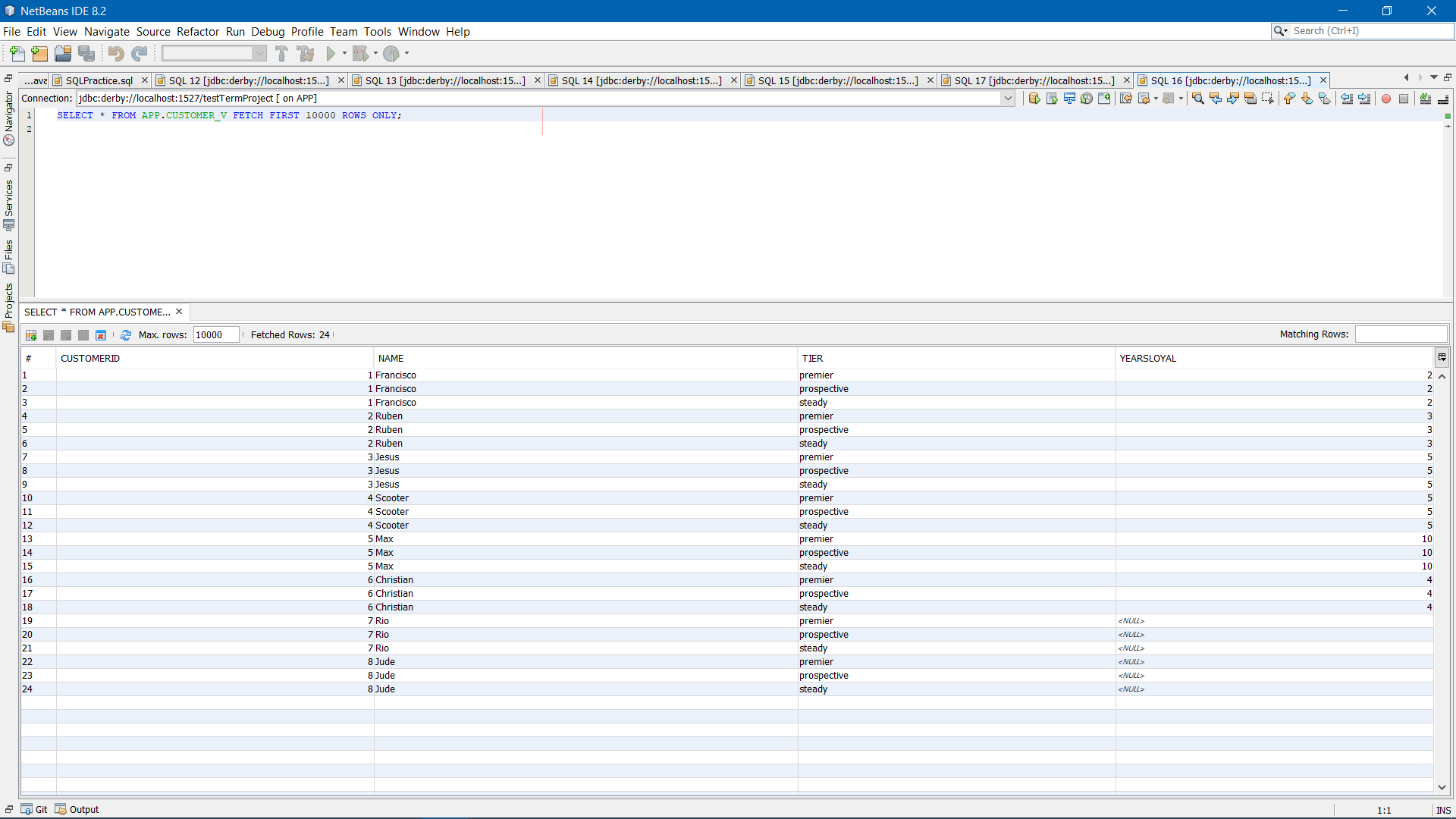
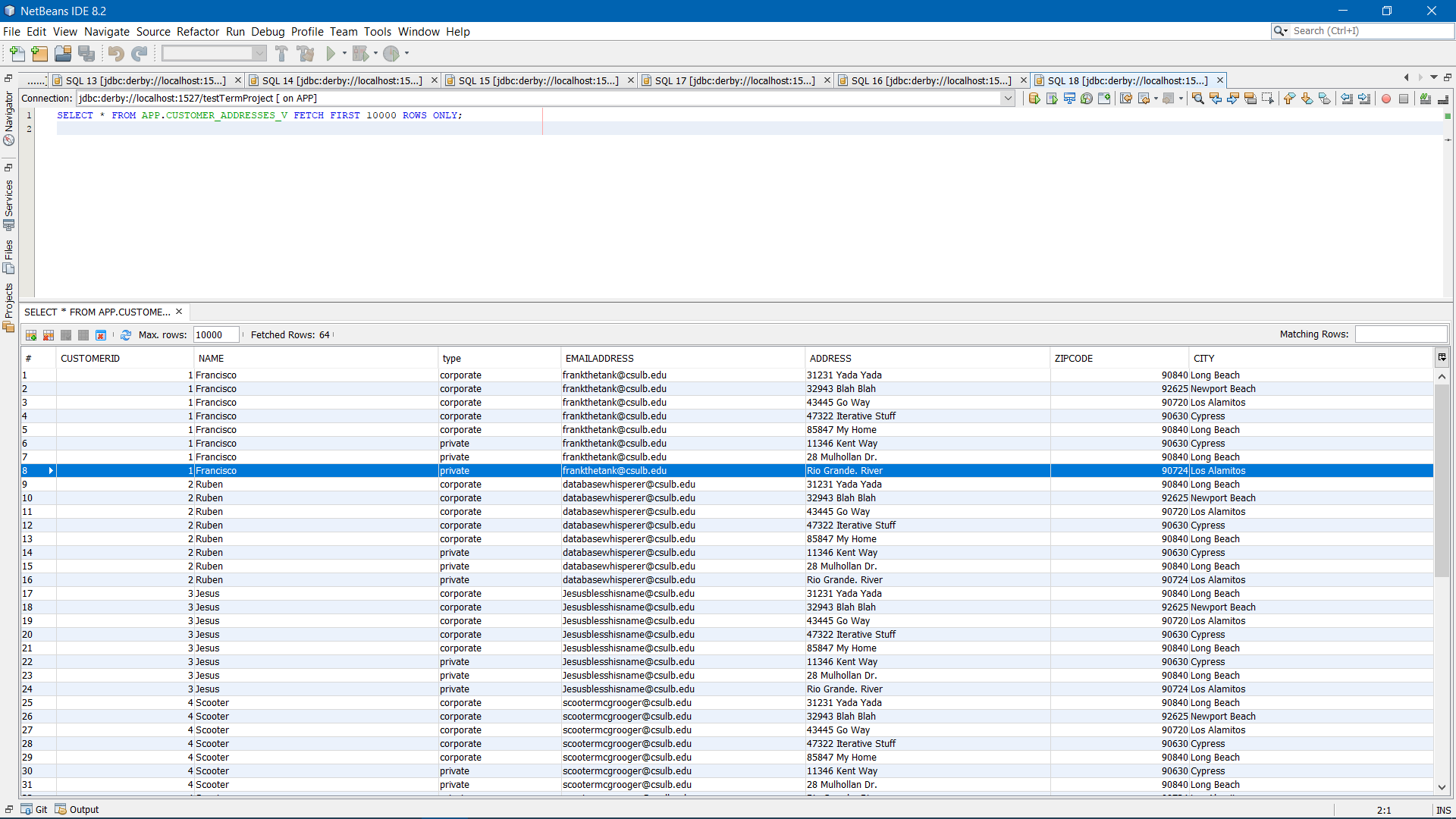
insert into steadyCustomers (steadyID, recentPayment) values

(1, 504.99), --Steady/Corp.

(2, 843.23), --Steady/Private

(5, 542.75); --Steady/Corp.

Views:

--------------------------VIEWS--------------------------------------------------------------------------  
--1--  
CREATE VIEW customer\_v AS  
 SELECT customerID, name, 'steady' AS tier, yearsLoyal  
 FROM customers NATURAL JOIN steadyCustomers  
 UNION  
 SELECT customerID, name, 'premier' AS tier, yearsLoyal  
 FROM customers NATURAL JOIN premierCustomers  
 UNION  
 SELECT customerID, name, 'prospective' AS tier, yearsLoyal  
 FROM customers NATURAL JOIN prospectiveCustomers;  
   
--2--  
CREATE VIEW customer\_addresses\_v AS  
 SELECT customerID, name, 'private' AS "type", address, zipcode, city  
 FROM customers   
 NATURAL JOIN privateCustomers  
 NATURAL JOIN privateAddresses  
 NATURAL JOIN zipLocations  
 UNION  
 SELECT customerID, name, 'corporate' AS "type", address, zipcode, city  
 FROM customers   
 NATURAL JOIN corporateCustomers  
 NATURAL JOIN corporateAddresses  
 NATURAL JOIN zipLocations;  
--3--  
CREATE VIEW mechanic\_mentor\_v AS  
 SELECT m1.name AS mentee, m2.name AS mentor  
 FROM   
 (SELECT name, mentorID FROM mechanics mc INNER JOIN mentorships mn  
 ON mc.mechanicID = mn.mechanicID) AS m1  
 INNER JOIN  
 (SELECT name, mentorID FROM mechanics mc INNER JOIN mentorships mn  
 ON mc.mechanicID = mn.mentorID) AS m2  
 ON m1.mentorID = m2.mentorID  
 ORDER BY mentee, mentor;  
   
--4--  
CREATE VIEW premier\_profits\_v AS  
 SELECT customerID, name, "year", fee AS "Premier Fee", steadycost AS "Steady Cost"  
 FROM   
 (SELECT customerID, sum(cost) AS steadycost  
 FROM customers  
 NATURAL JOIN vehicles  
 NATURAL JOIN invoices  
 NATURAL JOIN invoiceitems  
 NATURAL JOIN maintenanceitems  
 GROUP BY customerID, YEAR(cdate)  
 ) AS p  
 NATURAL JOIN customers  
 NATURAL JOIN premierCustomers  
 NATURAL JOIN annualfees;  
  
--5--  
CREATE VIEW prospective\_resurrection\_v AS  
 SELECT c.customerID, name, emailAddress  
 FROM customers c  
 INNER JOIN  
 (SELECT prospectiveID  
 FROM prospectivecustomers  
 NATURAL JOIN prospectivecustomeremails  
 GROUP BY prospectiveID  
 HAVING count(emailID) >= 3  
 ) AS ripeProspects  
 ON c.CUSTOMERID = ripeProspects.prospectiveID;

Queries

--Queries--  
  
--1--  
-- List the customers. For each customer, indicate which category he or she fall into, and his or her contact information.  
-- If you have more than one independent categorization of customers,  
-- please indicate which category the customer falls into for all of the categorizations.  
  
SELECT name, address, emailaddress, tier, "type"  
FROM customer\_v  
NATURAL JOIN customer\_addresses\_v;

--2--  
--For each service visit, list the total cost to the customer for that visit.  
  
Select customerID, name, Sum(cost)  
FROM customers  
NATURAL JOIN vehicles   
NATURAL JOIN Invoices   
Natural Join invoiceItems  
Natural Join MaintenanceItems  
Group by customerID,name, cdate;  
  
--3--  
--fix  
--List the top three customers in terms of their net spending for the past two years, --and the total that they have spent in that period.  
Select customerID, name, Sum(cost) as total  
from customers  
NATURAL JOIN Invoices   
Natural Join  
invoiceItems  
Natural Join   
MaintenaceItems  
where "date" between now() and DATE\_SUB(now, INTERVAL 2 YEAR)  
Group by customerID, name "date"  
Order by total desc   
limit 3;  
  
--4--  
SELECT mechanicID, name  
FROM mechanics  
NATURAL JOIN skillslist  
GROUP BY mechanicID, name  
HAVING count(skillname) > 2;  
  
--5--  
SELECT ms1.name, ms2.name  
FROM   
(SELECT \*  
 FROM skillsList s1  
 NATURAL JOIN mechanics m1) as ms1  
INNER JOIN  
(SELECT \*  
 FROM skillsList s2  
 NATURAL JOIN mechanics m2) AS ms2  
ON ms1.skillName = ms2.skillName  
WHERE ms1.mechanicID < ms2.mechanicID  
GROUP BY ms1.name, ms2.name  
HAVING count(ms2.skillName) > 2;  
  
--6--  
SELECT maintenanceID, itemname, total  
FROM   
(SELECT maintenanceID, sum(cost) AS total  
 FROM maintenancePackages  
 NATURAL JOIN maintenanceItems) AS t  
NATURAL JOIN maintenanceItems;  
  
--7--  
--fix  
--two new views--  
CREATE VIEW mechanic\_skills\_v AS  
SELECT mechanicID, name, skillName  
FROM mechanics  
NATURAL JOIN skillsList;  
  
CREATE VIEW item\_skills\_v AS  
SELECT itemname, skillName  
FROM maintenanceItems  
NATURAL JOIN skillsList;  
  
SELECT mechanicID, name, itemName, mechanic\_skills\_v.skillName  
FROM mechanics m  
CROSS JOIN maintenanceItems mi  
EXCEPT  
SELECT mechanicID, name, itemName, skillName  
FROM mechanic\_skills\_v  
INNER JOIN item\_skills\_v  
ON mechanic\_skills\_v.skillName = item\_skills\_v.skillName;  
  
--8--  
SELECT customerID, name, loyaltyPoints  
FROM customers  
ORDER BY loyaltyPoints DESC;  
  
--9--  
--fix  
--List the premier customers and the difference between what they have paid in the past year,  
--versus the services that they actually used during that same time.   
--List from the customers with the largest difference to the smallest.  
  
CREATE VIEW customer\_servicevalue\_v AS  
SELECT customerID, name, SUM(price) AS totalvalue  
 FROM customers  
 NATURAL JOIN vehicles  
 NATURAL JOIN invoiceitems  
 NATURAL JOIN maintenanceItems  
 WHERE "date" between now() and DATE\_SUB(now, INTERVAL 1 YEAR)  
 GROUP BY customerID, name;  
   
CREATE VIEW premiercustomer\_pay\_v AS  
SELECT customerID, name, SUM(amount) AS totalpay  
 FROM customers  
 NATURAL JOIN members  
 NATURAL JOIN premierCustomers  
 NATURAL JOIN payments  
 WHERE payDate between now() and DATE\_SUB(now, INTERVAL 1 YEAR)  
 GROUP BY customerID, name;  
   
SELECT customerID, name, ABS(totalpay - totalvalue) AS "difference"  
FROM customer\_servicescost\_vcost\_v  
NATURAL JOIN premiercustomer\_pay\_v;  
  
   
  
--10--  
--Report on the steady customers based on the net profit that we have made from them over the past year,  
-- and the dollar amount of that profit, in order from the greatest to the least  
CREATE VIEW customer\_pay\_v AS  
SELECT customerID, name, SUM(amount) AS totalpay  
 FROM customers  
 NATURAL JOIN members  
 NATURAL JOIN premierCustomers  
 NATURAL JOIN payments  
 WHERE payDate between now() and DATE\_SUB(now, INTERVAL 1 YEAR)  
 GROUP BY customerID, name;  
  
SELECT customerID, name, totalpay  
FROM customer\_pay\_v  
NATURAL JOIN steadyCustomers  
ORDER BY totalpay;  
  
  
  
--11--  
--List the three premier customers who have paid Dave’s Automotive  
--the greatest amount in the past year, and the sum of their payments over that period.  
--Be sure to take into account any discounts that they have earned by referring prospective customers.  
  
SELECT customerID, name, sum(amount) AS totalpay  
FROM premierCustomers  
NATURAL JOIN customers\_pay\_v  
GROUP BY customerID, name  
ORDER BY sum(amount) DESC  
LIMIT 3;  
  
  
--12--  
--List the five model, make, and year that have caused the most visits on average to Dave’s  
--automotive per vehicle in the past three years  
--along with the average number of visits per vehicle.  
  
SELECT model, make, year, AVG(visits) AS avgvisits  
FROM vehicles  
NATURAL JOIN  
(SELECT vehicleID, count(cDate) As visits  
 FROM invoices  
 WHERE cDate between now() and DATE\_SUB(now, INTERVAL 1 YEAR) AS v  
 GROUP BY invoiceID, vehicleID) as v  
GROUP BY model, make, year ;  
  
  
--13--  
--Find the mechanic who is mentoring the most other mechanics  
--List the skills that the mechanic is passing along to the other mechanics.  
  
CREATE VIEW mentoring\_V AS  
SELECT mentorID, name, mentoring  
FROM  
(SELECT mentorID, count(mechanicID) AS mentoring  
 FROM mentorships  
 GROUP BY mentorID) as m  
INNER JOIN mechanics m1  
ON m.mentorID = m1.mechanicID;  
  
SELECT \* FROM  
(SELECT \*FROM mentoring\_V  
WHERE mentoring =   
(SELECT MAX(mentoring)  
FROM mentoring\_V)) as m  
NATURAL JOIN skillsList;  
  
  
  
  
  
--14--  
--Find the three skills that have the fewest mechanics who have those skills.  
SELECT skillName, count(mechanicID) AS numofmechanics  
FROM skillList  
GROUP BY skillName  
ORDER BY numofmechanics;  
  
--15--  
--List the employees who are both service technicians as well as mechanics.  
SELECT mechanicID AS "Employee ID", name  
FROM mechanics  
INNER JOIN serviceTechnicians;  
  
--16--  
--Three additional queries that demonstrate the five additional business rules.   
--Feel free to create additional views to support these queries if you so desire.  
  
--Calculate cancellation fees  
SELECT customerID, name, ((fee - totalpay) \* .5) As cancellationfee  
FROM   
(SELECT customerID, name, SUM(amount) AS totalpay  
 FROM customers  
 NATURAL JOIN members  
 NATURAL JOIN premiercustomers  
 NATURAL JOIN payments  
 WHERE DATEPART(YEAR, payDate) = DATEPART(YEAR, now)  
 GROUP BY customerID  
) AS p;  
  
--display customers and their pets  
SELECT customerID, name, petName  
FROM customers  
NATURAL JOIN pets;  
  
--display donut eating stat  
SELECT customerID, name, count(dateEaten) AS "donuts eaten"  
FROM customers  
NATURAL JOIN donuts  
GROUP BY customerID, name;  
  
--display senior mechanic skill counts  
SELECT mechanic ID, name, count(skills)  
FROM mechanics  
NATURAL JOIN seniormechanics  
NATURAL JOIN skillList  
NATURAL JOIN skills  
GROUP BY mechanicID, name;

Business Rules:

1. Annual Contract has a cancellation fee.

2. 1 Free Donut for CorporateCustomers.

3. Customers are obligated to give the name and species of their pet(s).

4. Free 20 Loyalty Points for every Contract Renewal.

5. After 10 skills, a mechanic is promoted to a senior mechanic.