CAR RENTAL

Database Management System

Final Report

Course:	CPS 510 (Databases I)
Lab Section:	Section 5 (052)
Group Members:	Raghunadh Sai (500442386) Inheung Ji (500788269)

TABLE OF CONTENTS

Application Description

This Car Rental Database Management System is responsible for maintaining the relevant information necessary for a car rental business. The primary end users of our system are the employees who are going to be using our system to manage information relating to running a car rental business.

Our application will be able to provide employees with the ability to view information regarding:

- The cars that are available to be rented.
- The cars that are being rented presententły.
- The car models that are available to be rented.
- The locations from which rentals can take place.
- The current reservervations that are taking place.
- The past reservations that have happened.
- The billing status and information regarding all current and past reservations.
- The protections, coverages, accessories and services (i.e. insurance, loss damage, car seat, XM radio, GPS, etc.) that users will be able to select during the car reservation process.
- The customer and membership information of anyone that has created a reservation in our system.

Some of the basic functions that our system will be capable of doing are outlined below:

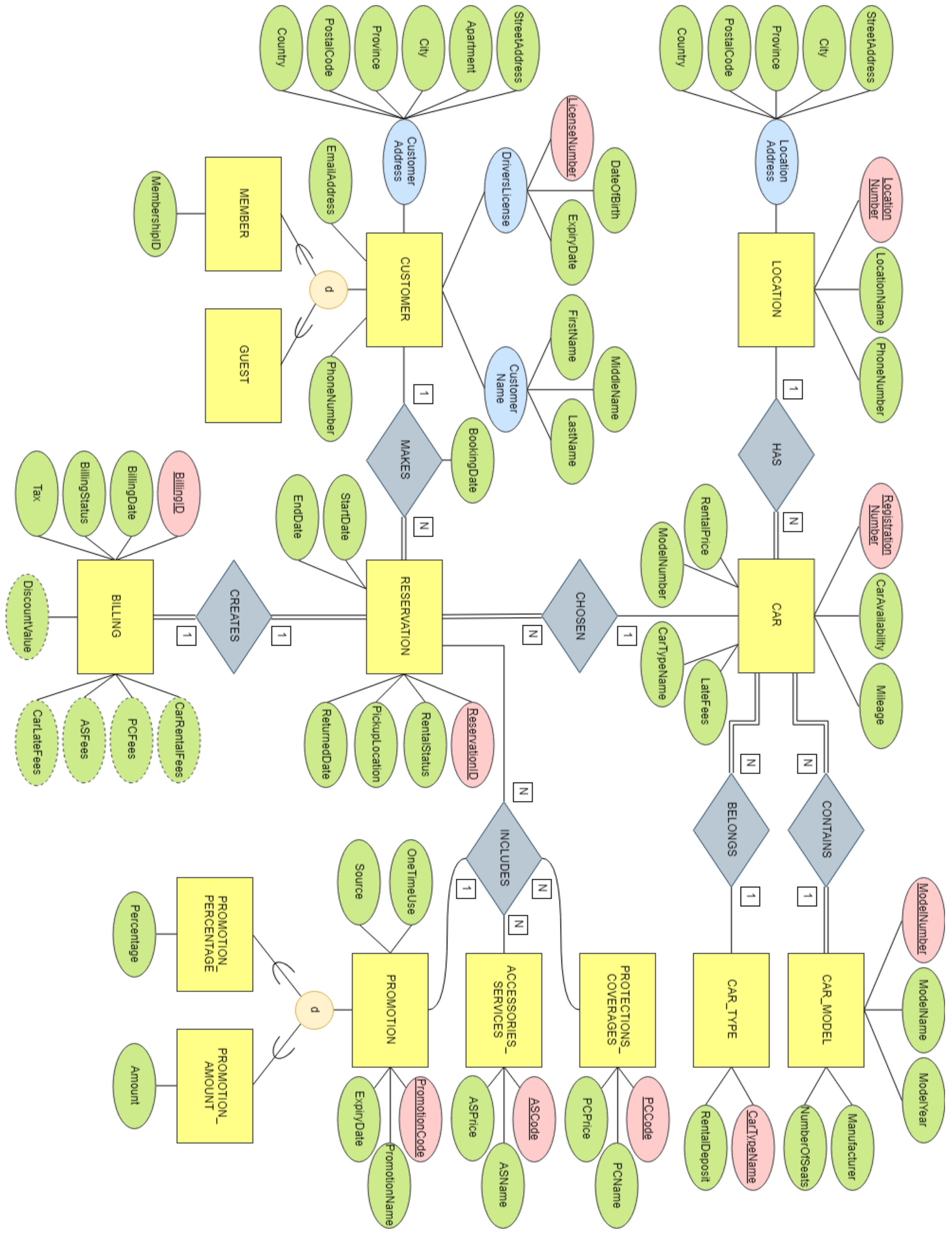
Function	Description
Insert New Customer	This function inserts information about a new customer into the database. The employee using the system can include the following information: (a) customer full name, (b) customer date of birth, (c) customer phone number, (d) customer email address, (e) customer home address and (f) customer drivers license information.
Remove Existing Customer	This function will remove the information about an existing customer within our database.
Create Car Rental Reservation	This function inserts a new car rental reservation based on customer information, car information (includes additional items selected by customer), and location information.
Add Promotions	This function inserts a new promotion offer that is available into the database. The employee using the system can provide the information regarding the promotion code, the promotion name and the value of the promotion.

Add Protections, Coverages, Accessories & Services	This function inserts new protections, coverages, accessories and services into the database. The employee using the system can provide the information regarding the code, the cost and the name of the item being added.
Update Promotions, Protections, Coversages & Services	The employee using the system will be able to update the information regarding the promotions, protections, coversages and services that is in the database.
Viewing Information About Different Aspects Of Database System	As mentioned above, the employee will be able to view the information regarding every aspect of the database aspect that is being stored. Please see bullet points above for more information.

In the next section, the ER diagram illustrates the different components of our database system.

- The yellow-coloured items represent the entities that exist in this database management system.
- The green-coloured items represent the attributes that each entity possesses.
- The red-coloured items represent the primary key of each entity.
- The grey-coloured items represent the relationships that exist between each entity.

ER Model


CUSTOMER
Customer Address
Country
PostalCode
Province
City
Apartment
StreetAddress
EmailAddress
PhoneNumber
DriversLicense
LicenseNumber
DateOfBirth
ExpiryDate
Customer Name
FirstName
MiddleName
LastName
MEMBER
MembershipID
GUEST
d
MAKES
BookingDate
1
N
LOCATION
Location Address
Country
PostalCode
Province
City
StreetAddress
Location Number
LocationName
PhoneNumber
HAS
1
N
CAR
Registration Number
CarAvailability
Mileage
RentalPrice
ModelNumber
CarTypeName
LateFees
CHOSEN
1
N
RESERVATION
StartDate
EndDate
ReturnedDate
PickupLocation
RentalStatus
ReservationID
CREATES
1
1
BILLING
BillingID
BillingDate
BillingStatus
Tax
DiscountValue
CarLateFees
ASFees
PCFees
CarRentalFees
INCLUDES
N
1
N
N
BELONGS
N
1
CONTAINS
N
1
PROMOTION
Source
OneTimeUse
ExpiryDate
PromotionCode
PromotionName
d
PROMOTION_PERCENTAGE
Percentage
PROMOTION_AMOUNT
Amount
ACCESSORIES_SERVICES
ASPrice
ASCode
ASName
PROTECTIONS_COVERAGES
PCPrice
PCCode
PCName
CAR_TYPE
RentalDeposit
CarTypeName
CAR_MODEL
ModelNumber
ModelName
ModelYear
NumberOfSeats
Manufacturer

Schema Design

1. Creating Tables

```
------------------------------------------------
-- Create Table Statements
------------------------------------------------

CREATE TABLE RESERVATION (
   ReservationID INTEGER PRIMARY KEY,
   RentalStatus VARCHAR2(3) CONSTRAINT reservation_check_out_in CHECK (RentalStatus
IN ('OUT', 'IN')),
   PickupLocation VARCHAR2(18) NOT NULL,
   StartDate DATE NOT NULL,
   EndDate DATE NOT NULL,
   ReturnedDate DATE
);

CREATE TABLE LOCATION (
   LocationNumber INTEGER PRIMARY KEY,
   LocationName VARCHAR2(18),
   PhoneNumber VARCHAR2(12),
   StreetAddress VARCHAR2(50) NOT NULL,
   City VARCHAR2(20) NOT NULL,
   Province VARCHAR2(2) DEFAULT 'ON',
   PostalCode VARCHAR2(7) NOT NULL,
   Country VARCHAR2(20) DEFAULT 'Canada'
);

CREATE TABLE CAR (
   RegistrationNumber INTEGER PRIMARY KEY,
   CarAvailability VARCHAR2(3) CONSTRAINT car_check_out_in CHECK (CarAvailability IN
('OUT', 'IN')),
   Mileage INTEGER,
   RentalPrice INTEGER NOT NULL,
   LateFees INTEGER,
   ModelNumber INTEGER NOT NULL,
   CarTypeName VARCHAR2(18)
);

CREATE TABLE CAR_MODEL (
   ModelNumber INTEGER PRIMARY KEY,
   ModelName VARCHAR2(18),
   ModelYear INTEGER,
   Manufacturer VARCHAR2(18),
   NumberOfSeats INTEGER
);

CREATE TABLE CAR_TYPE (
   CarTypeName VARCHAR2(18) PRIMARY KEY,
   RentalDeposit INTEGER
);

CREATE TABLE CUSTOMER (
   LicenseNumber VARCHAR2(18) PRIMARY KEY,
   ExpiryDate DATE NOT NULL,
   DateOfBirth DATE NOT NULL,
   FirstName VARCHAR2(20) NOT NULL,
   MiddleName VARCHAR2(1),
   LastName VARCHAR2(30) NOT NULL,
   EmailAddress VARCHAR2(50),
   PhoneNumber VARCHAR2(12) NOT NULL,
   StreetAddress VARCHAR2(50) NOT NULL,
   Apartment VARCHAR2(5),
   City VARCHAR2(20) NOT NULL,
   Province VARCHAR2(2) DEFAULT 'ON',
   PostalCode VARCHAR2(7) NOT NULL,
   Country VARCHAR2(20) DEFAULT 'Canada'
);

CREATE TABLE MEMBER (
   LicenseNumber VARCHAR2(18) REFERENCES CUSTOMER(LicenseNumber),
   MembershipID INTEGER
);

CREATE TABLE BILLING (
   BillingID INTEGER PRIMARY KEY,
   BillingStatus VARCHAR2(4) CONSTRAINT billing_status_check_good_bad CHECK
(BillingStatus IN ('GOOD', 'BAD')),
   BillingDate DATE NOT NULL,
   Tax INTEGER NOT NULL,
   CarRentalFees INTEGER NOT NULL,
   PCFees INTEGER,
   ASFees INTEGER,
   CarLateFees INTEGER,
   DiscountValue INTEGER
);

CREATE TABLE PROTECTIONS_COVERAGES (
   PCCode INTEGER PRIMARY KEY,
   PCName VARCHAR2(30),
   PCPrice INTEGER
);

CREATE TABLE ACCESSORIES_SERVICES (
   ASCode INTEGER PRIMARY KEY,
   ASName VARCHAR2(30),
   ASPrice INTEGER
```

```
);

CREATE TABLE PROMOTION (
   PromotionCode INTEGER PRIMARY KEY,
   PromotionName VARCHAR2(30),
   ExpiryDate DATE,
   OneTimeUse VARCHAR2(3) CONSTRAINT one_time_use_check_yes_no CHECK
(OneTimeUse IN ('YES', 'NO')),
   Source VARCHAR2(50)
);

CREATE TABLE PROMOTION_PERCENTAGE (
   PromotionCode INTEGER REFERENCES PROMOTION(PromotionCode),
   Percentage INTEGER,
   PRIMARY KEY(PromotionCode)
);

CREATE TABLE PROMOTION_AMOUNT (
   PromotionCode INTEGER REFERENCES PROMOTION(PromotionCode),
   Amount INTEGER,
   PRIMARY KEY(PromotionCode)
);

CREATE TABLE MAKES (
   LicenseNumber VARCHAR2(18) REFERENCES CUSTOMER(LicenseNumber),
   ReservationID INTEGER REFERENCES RESERVATION(ReservationID),
   BookingDate DATE,
   PRIMARY KEY(LicenseNumber,ReservationID)
);

CREATE TABLE CHOSEN (
   ReservationID INTEGER REFERENCES RESERVATION(ReservationID),
   RegistrationNumber INTEGER REFERENCES CAR(RegistrationNumber),
   PRIMARY KEY(ReservationID, RegistrationNumber)
);

CREATE TABLE HAS (
   LocationNumber INTEGER REFERENCES LOCATION(LocationNumber),
   RegistrationNumber INTEGER REFERENCES CAR(RegistrationNumber),
   PRIMARY KEY(LocationNumber, RegistrationNumber)
);

CREATE TABLE CONTAINS (
   RegistrationNumber INTEGER REFERENCES CAR(RegistrationNumber),
   ModelNumber INTEGER REFERENCES CAR_MODEL(ModelNumber),
   PRIMARY KEY(RegistrationNumber, ModelNumber)
);

CREATE TABLE BELONGS (
   RegistrationNumber INTEGER REFERENCES CAR(RegistrationNumber),
   CarTypeName VARCHAR2(18) REFERENCES CAR_TYPE(CarTypeName),
   PRIMARY KEY(RegistrationNumber, CarTypeName)
);

CREATE TABLE INCLUDES (
   ReservationID INTEGER REFERENCES RESERVATION(ReservationID),
   PCCode INTEGER REFERENCES PROTECTIONS_COVERAGES(PCCode),
   ASCode INTEGER REFERENCES ACCESSORIES_SERVICES(ASCode),
   PromotionCode INTEGER REFERENCES PROMOTION(PromotionCode),
   PRIMARY KEY(ReservationID, PCCode, ASCode, PromotionCode)
);

CREATE TABLE CREATES (
   ReservationID INTEGER REFERENCES RESERVATION(ReservationID),
   BillingID INTEGER REFERENCES BILLING(BillingID),
   PRIMARY KEY(ReservationID, BillingID)
);
```

2. Populating Tables With Data

```
------------------------------------------------
-- Adding In Values
------------------------------------------------

INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D3101-69581-63862', '2020/07/22', '1974/12/04', 'Paul', 'Arroyo', 'PaulDArroyo@dayrep.com',
'604-431-6663', '1037 Adelaide St', 'Toronto', 'M5H 1P6');
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D2345-34567-45678', '2022/09/12', '1993/01/10', 'Javid', 'Decster', 'javidD@gmail.com', '647-
222-2222', '100 Church St', 'Toronto', 'M2M 3B2');
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D4020-88182-19796', '2021/03/28', '1966/03/15', 'Katherine', 'Sisson',
'KatherineDSisson@armyspy.com', '519-570-6264', '4332 Water Street', 'Kitchener','N2H 5A5');
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D1416-89395-92248', '2024/07/20', '1992/11/10', 'Casey', 'Trejo',
'CaseyRTrejo@jourrapide.com', '519-473-1266', '3469 Hyde Park Road', 'London', 'N6H 3S2');
```

```
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D6960-19171-26137', '2024/11/21', '1995/12/08', 'Josephine', 'Mast',
'JosephineNMast@teleworm.us', '519-293-2149', '2991 Baker Street', 'London', 'N0N 0N0');
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D4348-32119-11009', '2024/05/27', '1990/08/03', 'William', 'Walter',
'WilliamDWalter@teleworm.us', '905-621-4183', '3830 Toy Avenue', 'Ajax', 'L1S 6L6');
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D9133-32635-40061', '2022/09/02', '1976/02/26', 'Tina', 'Baily', 'TinaABaily@teleworm.us', '905-
391-1159', '2605 Toy Avenue', 'Kitchener', 'L1W 3N9');
INSERT INTO CUSTOMER
(LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress,
PhoneNumber, StreetAddress, City, PostalCode)
VALUES
('D1601-90328-66848', '2024/09/12', '1994/10/30', 'Mattie', 'Warnock',
'MattieRWarnock@teleworm.us', '647-223-0149', '4093 Islington Ave', 'Toronto', 'M8V 3B6');

INSERT INTO MEMBER
(LicenseNumber, MembershipID)
VALUES
('D1601-90328-66848', 1001);
INSERT INTO MEMBER
(LicenseNumber, MembershipID)
VALUES
('D4348-32119-11009', 1002);
INSERT INTO MEMBER
(LicenseNumber, MembershipID)
VALUES
('D9133-32635-40061', 1003);
INSERT INTO MEMBER
(LicenseNumber, MembershipID)
VALUES
('D4020-88182-19796', 1004);
INSERT INTO MEMBER
(LicenseNumber, MembershipID)
VALUES
('D3101-69581-63862', 1005);

INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
```

```
VALUES
(716830987, 'IN', 148299, 120, 20, 18187118415, 'Truck');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(416341540, 'IN', 98213, 120, 20, 53517077183, 'Truck');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(448632952, 'IN', 119096, 120, 20, 14685153720, 'Truck');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(830323748, 'IN', 44556, 80, 20, 14040211286, 'Van');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(624749069, 'IN', 54522, 80, 20, 23552234653, 'Van');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(232940401, 'IN', 74632, 100, 20, 46568522408, 'Sedan');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(407177022, 'IN', 67354, 100, 20, 14332249770, 'Sedan');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(956503875, 'IN', 132452, 60, 10, 82381133464, 'Sedan');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(259930739, 'IN', 23456, 100, 20, 17210491613, 'Sedan');
INSERT INTO CAR
(RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber,
CarTypeName)
VALUES
(257923645, 'IN', 98765, 100, 20, 12760346714, 'Sedan');

INSERT INTO CAR_MODEL
```

```
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(18187118415, 'Ford Explorer', 2017, 'Ford', 7);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(53517077183, 'Ford F150', 2018, 'Ford', 5);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(14685153720, 'Jeep Wrangler', 2016, 'Jeep', 4);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(14040211286, 'Toyota Sienna', 2018, 'Toyota', 8);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(23552234653, 'Honda Odyssey', 2018, 'Honda', 8);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(46568522408, 'Honda Accord', 2018, 'Honda', 5);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(14332249770, 'Toyota Camry', 2018, 'Toyota', 5);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(82381133464, 'Honda Civic', 2017, 'Honda', 4);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(17210491613, 'Lexus GS', 2018, 'Lexus', 5);
INSERT INTO CAR_MODEL
(ModelNumber, ModelName, ModelYear, Manufacturer, NumberOfSeats)
VALUES
(12760346714, 'Toyota Corolla', 2017, 'Toyota', 5);

INSERT INTO CAR_TYPE
(CarTypeName, RentalDeposit)
VALUES
('Truck', 500);
INSERT INTO CAR_TYPE
(CarTypeName, RentalDeposit)
VALUES
('Van', 400);
INSERT INTO CAR_TYPE
```

```
(CarTypeName, RentalDeposit)
VALUES
('Sedan', 300);

INSERT INTO LOCATION
(LocationNumber, LocationName, PhoneNumber, StreetAddress, City, PostalCode)
VALUES
(5222, 'Billy Bishop', '416-777-1618', '161A Bay Street', 'Toronto', 'M5J 2S1');
INSERT INTO LOCATION
(LocationNumber, LocationName, PhoneNumber, StreetAddress, City, PostalCode)
VALUES
(2498, 'Hamilton', '905-525-1400', '237 Main Street East', 'Hamilton', 'L8N 1H4');
INSERT INTO LOCATION
(LocationNumber, LocationName, PhoneNumber, StreetAddress, City, PostalCode)
VALUES
(8331, 'Toronto Pearson', '905-676-1100', '5990 Airport Road', 'Toronto', 'L5P 1B2');
INSERT INTO LOCATION
(LocationNumber, LocationName, PhoneNumber, StreetAddress, City, PostalCode)
VALUES
(7670, 'London Airport', '519-451-8400', '1750 Crumlin Road', 'London', 'N5V 3B6');

INSERT INTO RESERVATION
(ReservationID, RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate)
VALUES
(1, 'IN', '8331', '2018/09/28', '2018/09/29', '2018/09/29');
INSERT INTO RESERVATION
(ReservationID, RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate)
VALUES
(2, 'IN', '8331', '2018/07/02', '2018/09/06', '2018/09/06');
INSERT INTO RESERVATION
(ReservationID, RentalStatus, PickupLocation, StartDate, EndDate)
VALUES
(3, 'OUT', '8331', '2018/10/01', '2018/10/10');
INSERT INTO RESERVATION
(ReservationID, RentalStatus, PickupLocation, StartDate, EndDate)
VALUES
(4, 'OUT', '5222', '2018/10/04', '2018/10/06');
INSERT INTO RESERVATION
(ReservationID, RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate)
VALUES
(5, 'IN', '5222', '2018/10/03', '2018/10/03', '2018/10/04');

INSERT INTO PROTECTIONS_COVERAGES
(PCCode, PCName, PCPrice)
VALUES
(101, 'Loss Damage Waiver', 40);
INSERT INTO PROTECTIONS_COVERAGES
(PCCode, PCName, PCPrice)
VALUES
```

```
(102, 'Personal Accident Insurance', 10);
INSERT INTO PROTECTIONS_COVERAGES
(PCCode, PCName, PCPrice)
VALUES
(103, 'Personal Effects Protection', 5);

INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice)
VALUES
(201, 'Additional Driver', 10);
INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice)
VALUES
(202, 'Travel Tab', 25);
INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice)
VALUES
(203, 'GPS', 10);
INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice)
VALUES
(204, 'Extended Roadside Assistance', 10);
INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice)
VALUES
(205, 'Child Safety Seat', 5);

INSERT INTO PROMOTION
(PromotionCode, PromotionName, OneTimeUse)
VALUES
(301, 'First Rental', 'YES');
INSERT INTO PROMOTION
(PromotionCode, PromotionName, OneTimeUse)
VALUES
(302, '10x Rental', 'NO');
INSERT INTO PROMOTION
(PromotionCode, PromotionName, ExpiryDate, OneTimeUse, Source)
VALUES
(649052, 'Christmas', '2018/12/31', 'YES', 'Online');

INSERT INTO PROMOTION_PERCENTAGE
(PromotionCode, Percentage)
VALUES
(649052, 10);

INSERT INTO PROMOTION_AMOUNT
(PromotionCode, Amount)
VALUES
(301, 100);
```

```
INSERT INTO PROMOTION_AMOUNT
(PromotionCode, Amount)
VALUES
(302, 50);

INSERT INTO BILLING
(BillingID, BillingStatus, BillingDate, Tax, CarRentalFees, PCFees, ASFees, DiscountValue)
VALUES
(1, 'GOOD', '2018/09/29', 13, 200, 80, 10, 100);
INSERT INTO BILLING
(BillingID, BillingStatus, BillingDate, Tax, CarRentalFees)
VALUES
(2, 'GOOD', '2018/09/06', 13, 3600);
INSERT INTO BILLING
(BillingID, BillingStatus, BillingDate, Tax, CarRentalFees, PCFees, ASFees, DiscountValue)
VALUES
(3, 'BAD', '2018/10/01', 13, 800, 400, 100, 10);
INSERT INTO BILLING
(BillingID, BillingStatus, BillingDate, Tax, CarRentalFees, PCFees, ASFees)
VALUES
(4, 'BAD', '2018/10/04', 13, 360, 30, 30);
INSERT INTO BILLING
(BillingID, BillingStatus, BillingDate, Tax, CarRentalFees, CarLateFees)
VALUES
(5, 'GOOD', '2018/10/04', 13, 240, 20);

INSERT INTO MAKES
(LicenseNumber, ReservationID, BookingDate)
VALUES
('D3101-69581-63862', 1, '2018/09/27');
INSERT INTO MAKES
(LicenseNumber, ReservationID, BookingDate)
VALUES
('D2345-34567-45678', 2, '2018/07/01');
INSERT INTO MAKES
(LicenseNumber, ReservationID, BookingDate)
VALUES
('D4020-88182-19796', 3, '2018/09/28');
INSERT INTO MAKES
(LicenseNumber, ReservationID, BookingDate)
VALUES
('D1416-89395-92248', 4, '2018/10/03');
INSERT INTO MAKES
(LicenseNumber, ReservationID, BookingDate)
VALUES
('D6960-19171-26137', 5, '2018/10/02');

INSERT INTO CHOSEN
(ReservationID, RegistrationNumber)
```

```
VALUES
(1, 716830987);
INSERT INTO CHOSEN
(ReservationID, RegistrationNumber)
VALUES
(2, 956503875);
INSERT INTO CHOSEN
(ReservationID, RegistrationNumber)
VALUES
(3, 407177022);
INSERT INTO CHOSEN
(ReservationID, RegistrationNumber)
VALUES
(4, 232940401);
INSERT INTO CHOSEN
(ReservationID, RegistrationNumber)
VALUES
(5, 448632952);

INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(5222, 716830987);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(2498, 416341540);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(8331, 448632952);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(7670, 830323748);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(5222, 624749069);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(2498, 232940401);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(8331, 407177022);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
```

```
VALUES
(7670, 956503875);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(7670, 259930739);
INSERT INTO HAS
(LocationNumber, RegistrationNumber)
VALUES
(8331, 257923645);

INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(716830987, 18187118415);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(416341540, 53517077183);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(448632952, 14685153720);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(830323748, 14040211286);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(624749069, 23552234653);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(232940401, 46568522408);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(407177022, 14332249770);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(956503875, 82381133464);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
VALUES
(259930739, 17210491613);
INSERT INTO CONTAINS
(RegistrationNumber, ModelNumber)
```

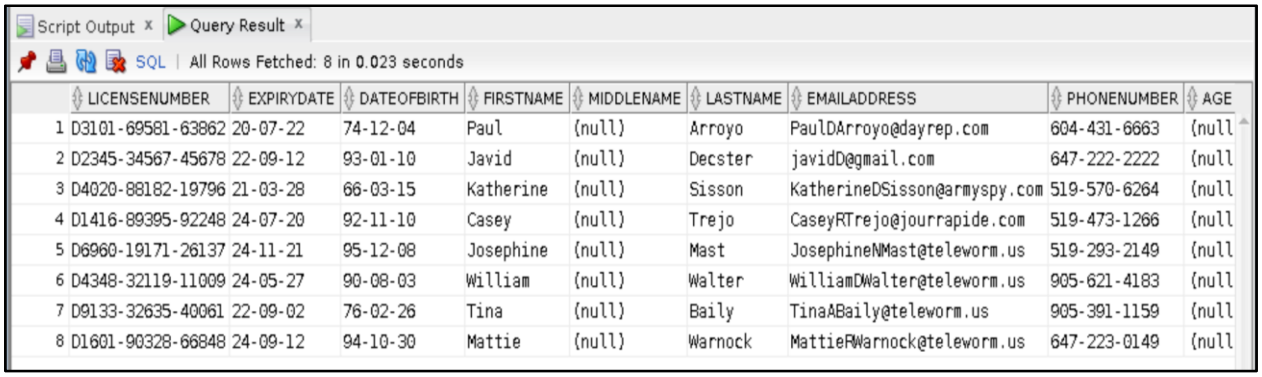
```
VALUES
(257923645, 12760346714);

INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(716830987, 'Truck');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(416341540, 'Truck');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(448632952, 'Truck');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(830323748, 'Van');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(624749069, 'Van');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(232940401, 'Sedan');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(407177022, 'Sedan');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(956503875, 'Sedan');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(259930739, 'Sedan');
INSERT INTO BELONGS
(RegistrationNumber, CarTypeName)
VALUES
(257923645, 'Sedan');

INSERT INTO INCLUDES
(ReservationID, PCCode, ASCode, PromotionCode)
VALUES
(1, 101, 205, 301);
INSERT INTO INCLUDES
```

```
(ReservationID, PCCode, ASCode, PromotionCode)
VALUES
(2, 101, 201, 301);
INSERT INTO INCLUDES
(ReservationID, PCCode, ASCode, PromotionCode)
VALUES
(3, 101, 202, 649052);
INSERT INTO INCLUDES
(ReservationID, PCCode, ASCode, PromotionCode)
VALUES
(4, 102, 203, 649052);
INSERT INTO INCLUDES
(ReservationID, PCCode, ASCode, PromotionCode)
VALUES
(5, 103, 204, 301);

INSERT INTO CREATES
(ReservationID, BillingID)
VALUES
(1, 1);
INSERT INTO CREATES
(ReservationID, BillingID)
VALUES
(2, 2);
INSERT INTO CREATES
(ReservationID, BillingID)
VALUES
(3, 3);
INSERT INTO CREATES
(ReservationID, BillingID)
VALUES
(4, 4);
INSERT INTO CREATES
(ReservationID, BillingID)
VALUES
(5, 5);
```

Simple Queries & Views

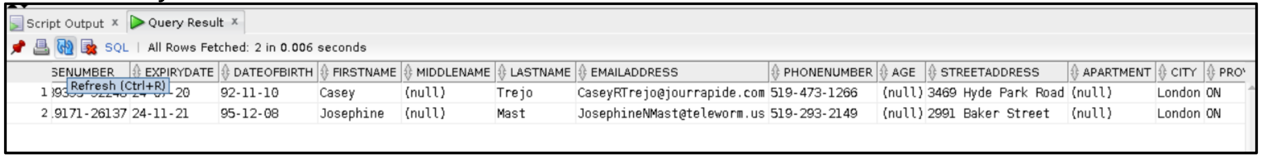
1. **CUSTOMER** Queries

SELECT *
FROM CUSTOMER



```
Script Output ×   Query Result ×
SQL | All Rows Fetched: 8 in 0.023 seconds

  LICENSENUMBER     EXPIRYDATE DATEOFBIRTH FIRSTNAME MIDDLENAME LASTNAME EMAILADDRESS                PHONENUMBER  AGE
1 D3101-69581-63862 20-07-22   74-12-04    Paul      (null)     Arroyo   PaulDArroyo@dayrep.com      604-431-6663 (null
2 D2345-34567-45678 22-09-12   93-01-10    Javid     (null)     Decster  javidD@gmail.com            647-222-2222 (null
3 D4020-88182-19796 21-03-28   66-03-15    Katherine (null)     Sisson   KatherineDSisson@armyspy.com 519-570-6264 (null
4 D1416-89395-92248 24-07-20   92-11-10    Casey     (null)     Trejo    CaseyRTrejo@jourrapide.com  519-473-1266 (null
5 D6960-19171-26137 24-11-21   95-12-08    Josephine (null)     Mast     JosephineNMast@teleworm.us  519-293-2149 (null
6 D4348-32119-11009 24-05-27   90-08-03    William   (null)     Walter   WilliamDWalter@teleworm.us  905-621-4183 (null
7 D9133-32635-40061 22-09-02   76-02-26    Tina      (null)     Baily    TinaABaily@teleworm.us      905-391-1159 (null
8 D1601-90328-66848 24-09-12   94-10-30    Mattie    (null)     Warnock  MattieRWarnock@teleworm.us  647-223-0149 (null
```

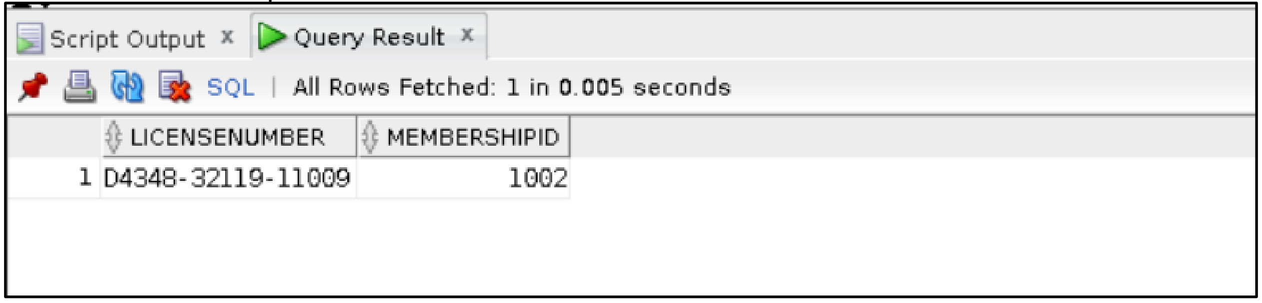
SELECT *
FROM CUSTOMER
WHERE City = 'London'



```
Script Output ×   Query Result ×
SQL | All Rows Fetched: 2 in 0.006 seconds

  SENUMBER     EXPIRYDATE DATEOFBIRTH FIRSTNAME MIDDLENAME LASTNAME EMAILADDRESS               PHONENUMBER  AGE    STREETADDRESS        APARTMENT CITY   PRO
1 [illegible]             92-11-10    Casey     (null)     Trejo    CaseyRTrejo@jourrapide.com 519-473-1266 (null) 3469 Hyde Park Road  (null)    London ON
2 9171-26137   24-11-21   95-12-08    Josephine (null)     Mast     JosephineNMast@teleworm.us 519-293-2149 (null) 2991 Baker Street    (null)    London ON
```

2. **MEMBER** Queries

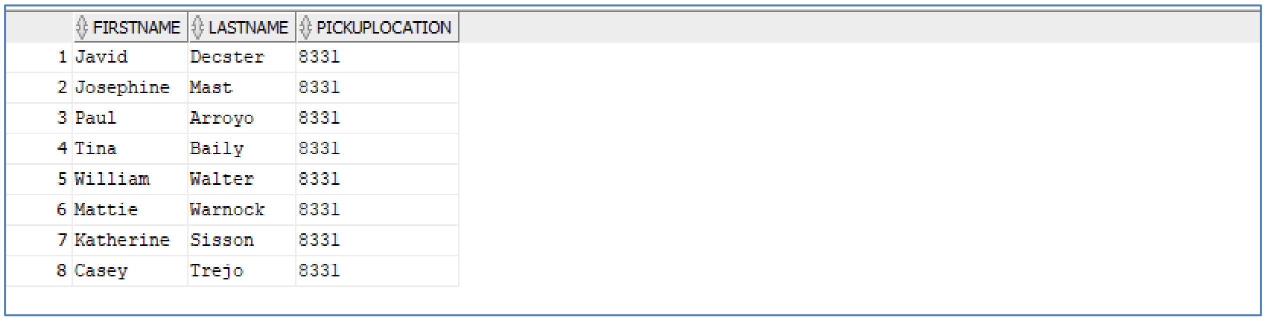
SELECT *
FROM MEMBER
WHERE MembershipID = 1002



```
Script Output ×   Query Result ×
SQL | All Rows Fetched: 1 in 0.005 seconds

  LICENSENUMBER     MEMBERSHIPID
1 D4348-32119-11009         1002
```

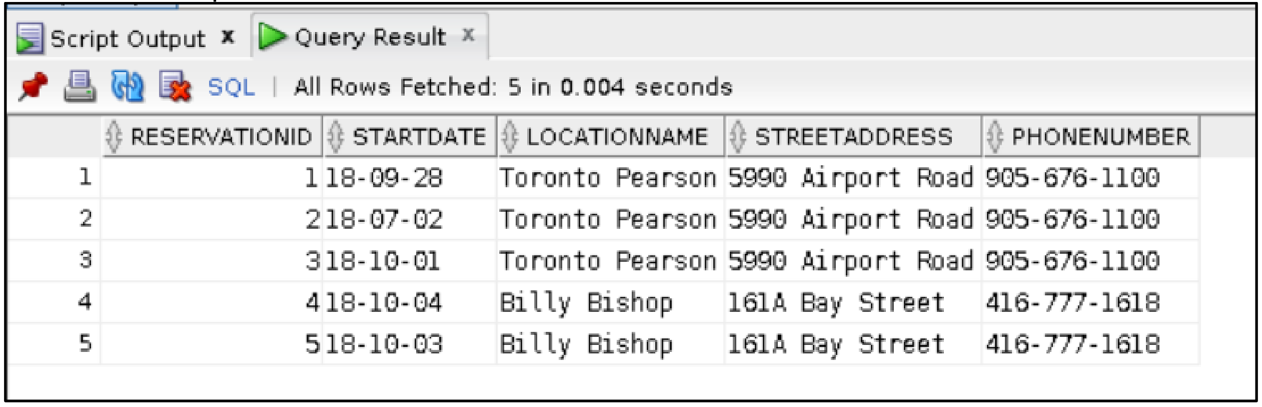
3. **RESERVATION** Queries

```
SELECT distinct c.FirstName, c.LastName, r.PickupLocation
FROM RESERVATION r, CUSTOMER c
WHERE EXISTS
(Select  *
FROM RESERVATION rv
WHERE rv.RentalStatus = 'OUT')
and r.PICKUPLOCATION = 8331
```



	FIRSTNAME	LASTNAME	PICKUPLOCATION
1	Javid	Decster	8331
2	Josephine	Mast	8331
3	Paul	Arroyo	8331
4	Tina	Baily	8331
5	William	Walter	8331
6	Mattie	Warnock	8331
7	Katherine	Sisson	8331
8	Casey	Trejo	8331

```
SELECT ReservationID, StartDate, LocationName, StreetAddress, PhoneNumber
FROM RESERVATION r, LOCATION l
WHERE r.PickupLocation = l.LocationNumber
```

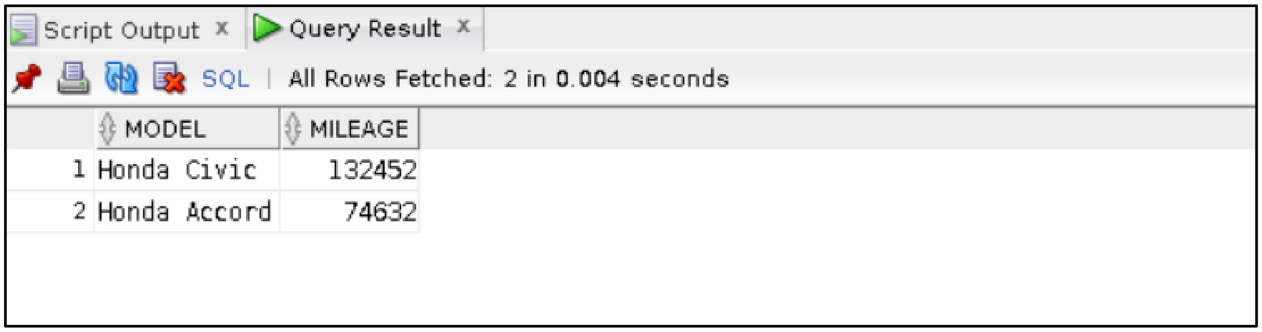


All Rows Fetched: 5 in 0.004 seconds

	RESERVATIONID	STARTDATE	LOCATIONNAME	STREETADDRESS	PHONENUMBER
1	1	18-09-28	Toronto Pearson	5990 Airport Road	905-676-1100
2	2	18-07-02	Toronto Pearson	5990 Airport Road	905-676-1100
3	3	18-10-01	Toronto Pearson	5990 Airport Road	905-676-1100
4	4	18-10-04	Billy Bishop	161A Bay Street	416-777-1618
5	5	18-10-03	Billy Bishop	161A Bay Street	416-777-1618

4. CAR Queries

```
SELECT cm.modelname, car.mileage
FROM CAR car, car_model cm, RESERVATION reservation, chosen
WHERE
        cm.manufacturer = 'Honda' AND
        car.modelnumber = cm.modelnumber AND
        car.registrationnumber = chosen.registrationnumber AND
        chosen.reservationid = reservation.reservationid
```

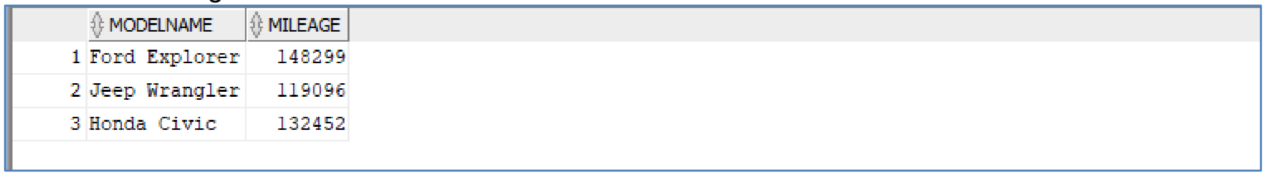


All Rows Fetched: 2 in 0.004 seconds

	MODEL	MILEAGE
1	Honda Civic	132452
2	Honda Accord	74632

5. CAR_MODEL Queries

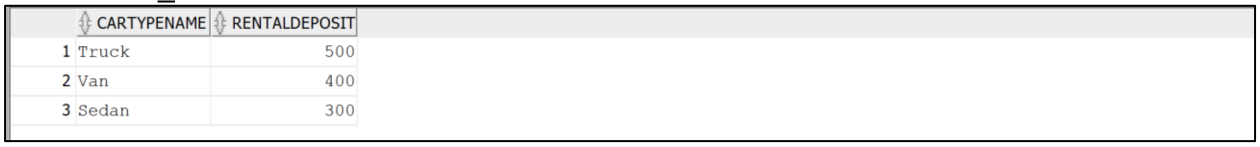
```
SELECT cm.modelname, Mileage
FROM car_model cm, CAR ca, car_type ct
WHERE
        ct.cartypename = 'Truck' AND
        cm.modelnumber = ca.modelnumber AND NOT
        ca.mileage < 100000
```



	MODELNAME	MILEAGE
1	Ford Explorer	148299
2	Jeep Wrangler	119096
3	Honda Civic	132452

6. CAR_TYPE Queries

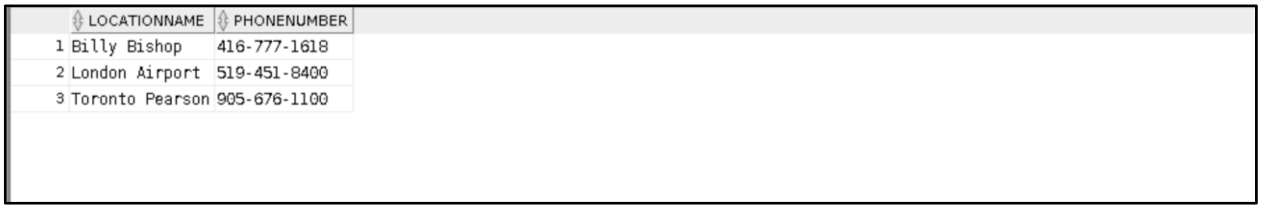
```
SELECT *
FROM CAR_TYPE
```



	CARTYPENAME	RENTALDEPOSIT
1	Truck	500
2	Van	400
3	Sedan	300

7. LOCATION Queries

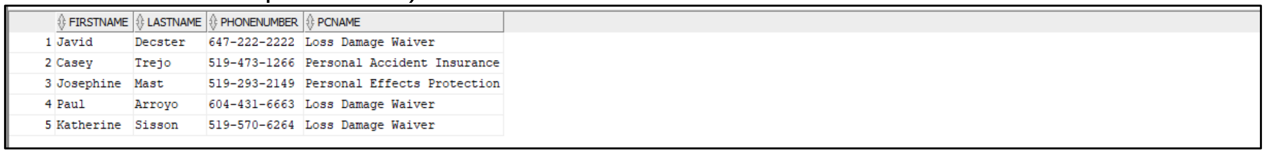
```
SELECT distinct LocationName, PhoneNumber
FROM LOCATION l , HAS h
WHERE l.LocationNumber = h.LocationNumber
        And h.LocationNumber >= 5000
```



	LOCATIONNAME	PHONENUMBER
1	Billy Bishop	416-777-1618
2	London Airport	519-451-8400
3	Toronto Pearson	905-676-1100

8. **PROTECTIONS_COVERAGES** Queries

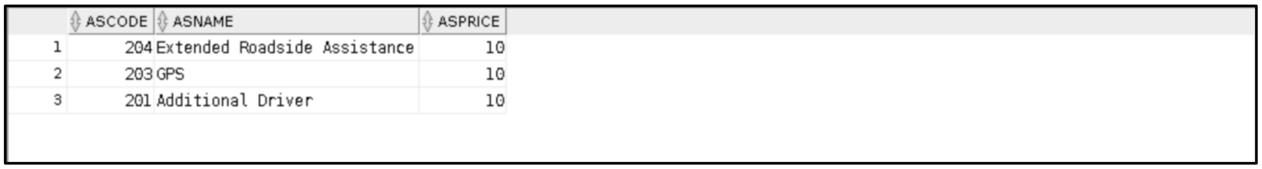
```
SELECT DISTINCT c.FirstName, c.LastName ,c.PhoneNumber, pc.pcName
FROM CUSTOMER c, RESERVATION r, INCLUDES i, PROTECTIONS_COVERAGES pc,
makes m
WHERE   i.reservationid = r.reservationid
 AND i.pccode = pc.pccode
 AND m.reservationid = r.reservationid
 AND m.licensenumber = c.licensenumber
AND NOT EXISTS
(SELECT *
 FROM MEMBER
WHERE MembershipID = 1006)
```



	FIRSTNAME	LASTNAME	PHONENUMBER	PCNAME
1	Javid	Decster	647-222-2222	Loss Damage Waiver
2	Casey	Trejo	519-473-1266	Personal Accident Insurance
3	Josephine	Mast	519-293-2149	Personal Effects Protection
4	Paul	Arroyo	604-431-6663	Loss Damage Waiver
5	Katherine	Sisson	519-570-6264	Loss Damage Waiver

9. **ACCESSORIES_SERVICES** Queries

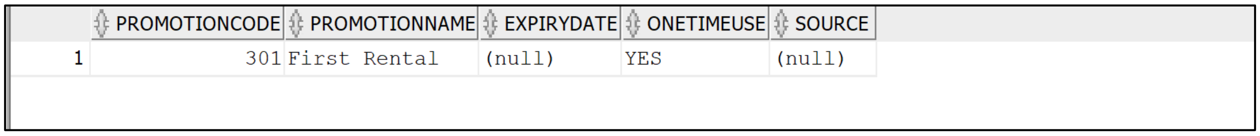
```
SELECT *
FROM ACCESSORIES_SERVICES
WHERE ASPrice = 10
ORDER BY AsCode DESC
```



	ASCODE	ASNAME	ASPRICE
1	204	Extended Roadside Assistance	10
2	203	GPS	10
3	201	Additional Driver	10

10. **PROMOTION** Queries

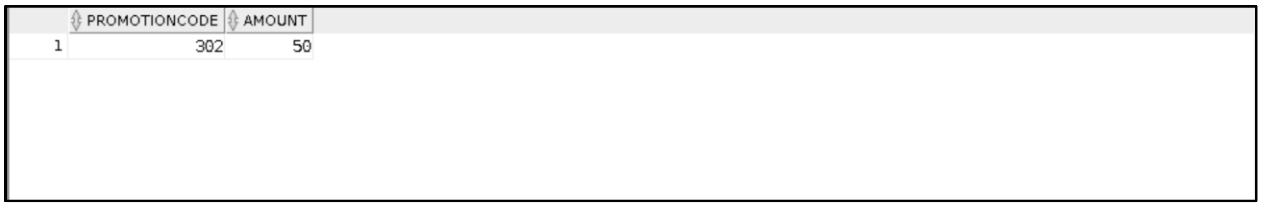
```
SELECT *
FROM PROMOTION
WHERE PromotionName = 'First Rental'
```



	PROMOTIONCODE	PROMOTIONNAME	EXPIRYDATE	ONETIMEUSE	SOURCE
1	301	First Rental	(null)	YES	(null)

11. **PROMOTION_AMOUNT** Queries

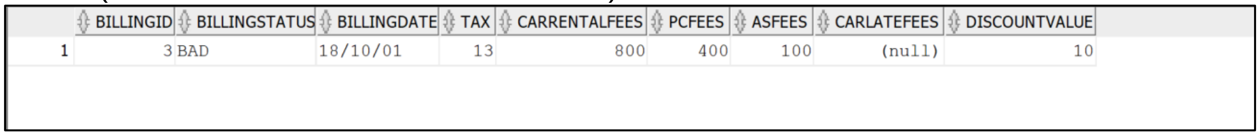
```
SELECT *
FROM PROMOTION_AMOUNT
WHERE Amount <= 70
```



	PROMOTIONCODE	AMOUNT
1	302	50

12. **BILLING** Queries

```
SELECT *
from BILLING
WHERE (BillingStatus = 'BAD' and NOT(DiscountValue = Null))
        OR
        (carrentalfees >=300 and AsFees >=50)
```

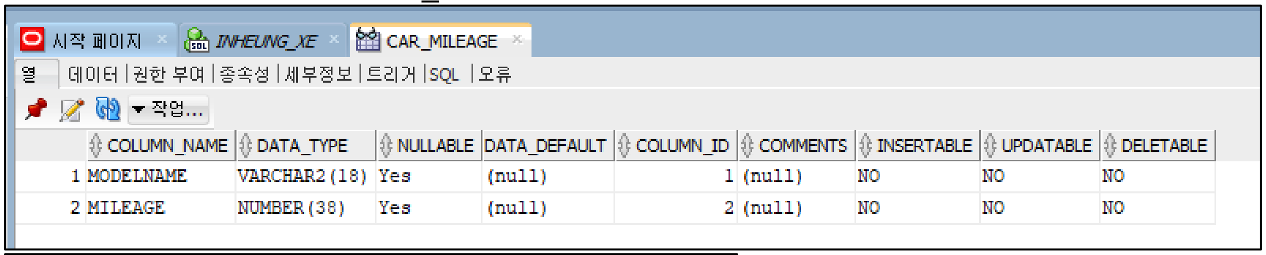


	BILLINGID	BILLINGSTATUS	BILLINGDATE	TAX	CARRENTALFEES	PCFEES	ASFEES	CARLATEFEES	DISCOUNTVALUE
1	3	BAD	18/10/01	13	800	400	100	(null)	10

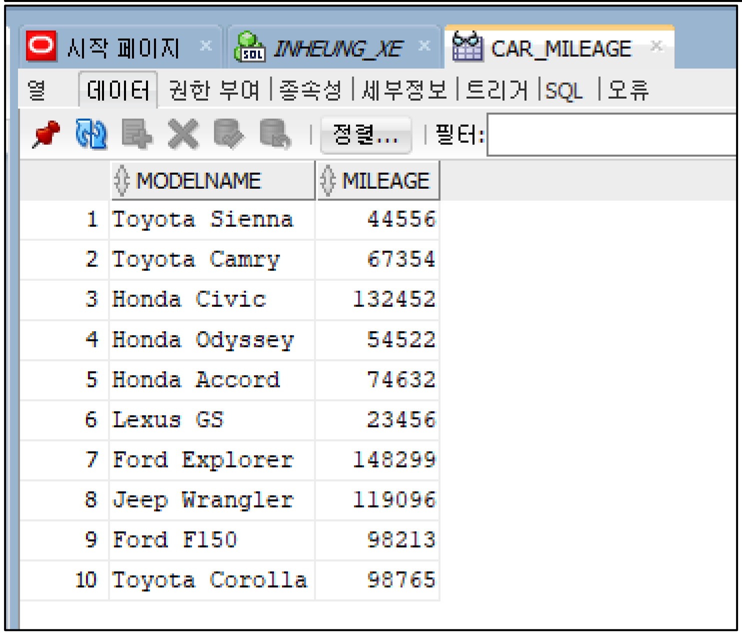
CREATING VIEWS

1. Mileage View

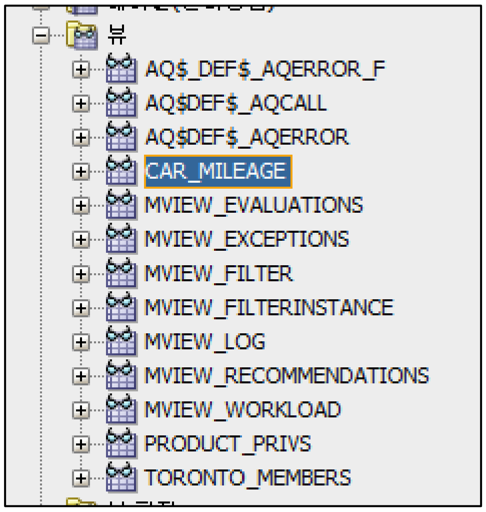
```
CREATE VIEW Car_Mileage
AS SELECT DISTINCT car_model.modelname, car.mileage
FROM car, car_model
WHERE car.modelnumber = car_model.modelnumber
```


시작 페이지
INHEUNG_XE
CAR_MILEAGE
열 데이터 | 권한 부여 | 종속성 | 세부정보 | 트리거 | SQL | 오류
작업...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS	INSERTABLE	UPDATABLE	DELETABLE
1	MODELNAME	VARCHAR2(18)	Yes	(null)	1	(null)	NO	NO	NO
2	MILEAGE	NUMBER(38)	Yes	(null)	2	(null)	NO	NO	NO

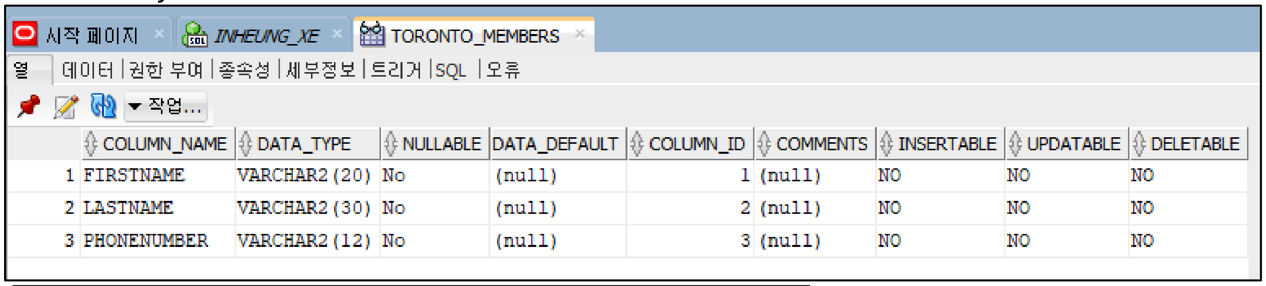

시작 페이지
INHEUNG_XE
CAR_MILEAGE
열 데이터 | 권한 부여 | 종속성 | 세부정보 | 트리거 | SQL | 오류
정렬...
필터:

	MODELNAME	MILEAGE
1	Toyota Sienna	44556
2	Toyota Camry	67354
3	Honda Civic	132452
4	Honda Odyssey	54522
5	Honda Accord	74632
6	Lexus GS	23456
7	Ford Explorer	148299
8	Jeep Wrangler	119096
9	Ford F150	98213
10	Toyota Corolla	98765

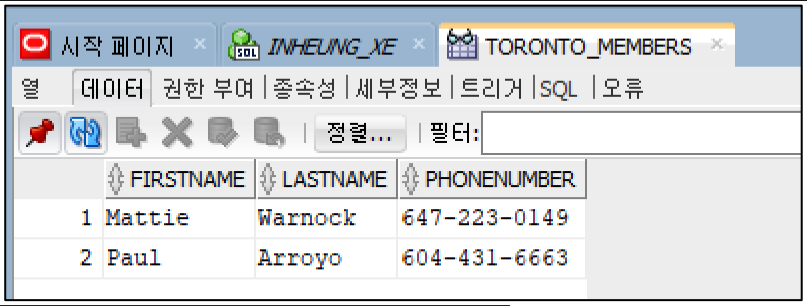


2. “Members In Toronto” View

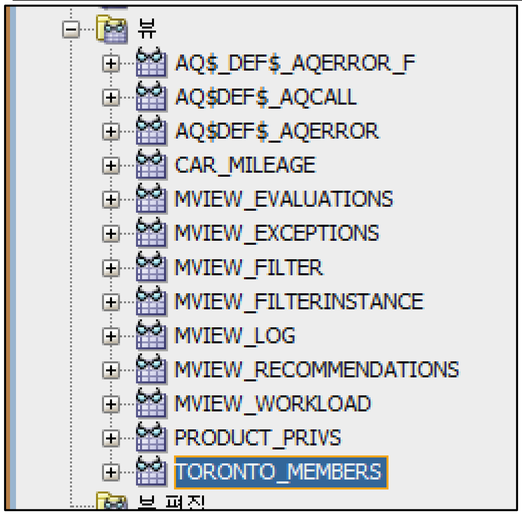
```
CREATE VIEW Toronto_Members
AS SELECT FirstName, LastName, PhoneNumber
from CUSTOMER c, MEMBER m
WHERE
        c.LicenseNumber = m.LicenseNumber AND
        c.City = 'Toronto'
```



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS	INSERTABLE	UPDATABLE	DELETABLE
1	FIRSTNAME	VARCHAR2(20)	No	(null)	1	(null)	NO	NO	NO
2	LASTNAME	VARCHAR2(30)	No	(null)	2	(null)	NO	NO	NO
3	PHONENUMBER	VARCHAR2(12)	No	(null)	3	(null)	NO	NO	NO



	FIRSTNAME	LASTNAME	PHONENUMBER
1	Mattie	Warnock	647-223-0149
2	Paul	Arroyo	604-431-6663

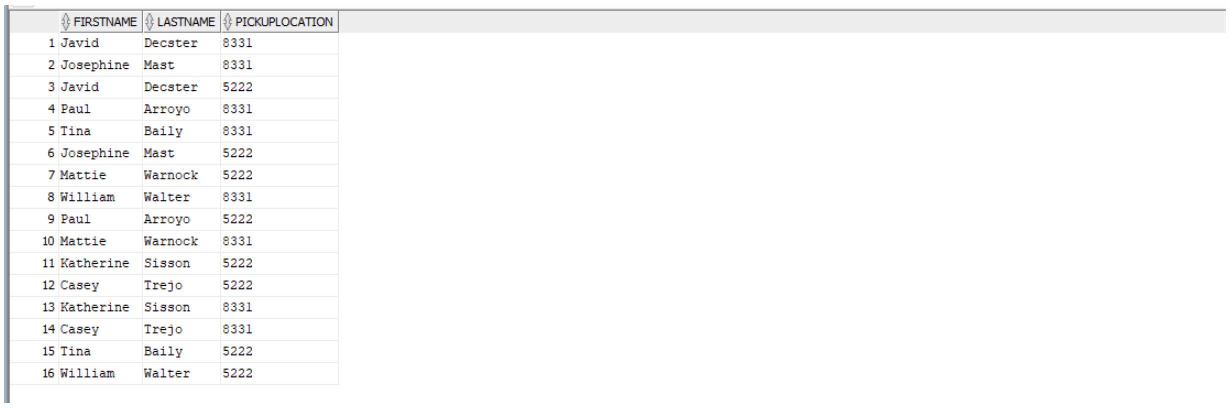


Advanced Queries

1. Using EXISTS

List all customers’ first name, last name, and pick-up location included customers’ billing status is good and pickup location is 8331.

```
SELECT DISTINCT c.FirstName, c.LastName, r.PickupLocation
FROM RESERVATION r, CUSTOMER c
WHERE EXISTS
(Select  cu.FIRSTNAME, cu.LASTNAME
FROM BILLING b, MAKES ma, CUSTOMER cu, RESERVATION rv, CREATES cr
WHERE ma.RESERVATIONID = rv.RESERVATIONID
AND rv.RESERVATIONID = cr.RESERVATIONID
AND cr.BILLINGID = b.BILLINGID
AND ma.LICENSENUMBER = cu.LICENSENUMBER
AND b.BILLINGSTATUS = 'GOOD'
AND rv.PICKUPLOCATION = '8331');
```

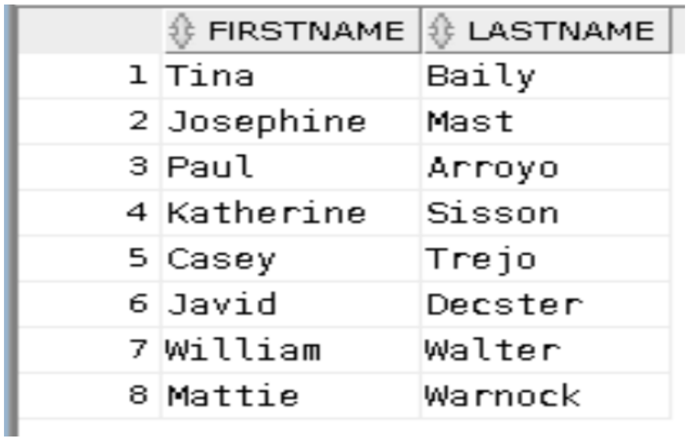


```
     FIRSTNAME  LASTNAME  PICKUPLOCATION
   1 Javid      Decster   8331
   2 Josephine  Mast      8331
   3 Javid      Decster   5222
   4 Paul       Arroyo    8331
   5 Tina       Baily     8331
   6 Josephine  Mast      5222
   7 Mattie     Warnock   5222
   8 William    Walter    8331
   9 Paul       Arroyo    5222
  10 Mattie     Warnock   8331
  11 Katherine  Sisson    5222
  12 Casey      Trejo     5222
  13 Katherine  Sisson    8331
  14 Casey      Trejo     8331
  15 Tina       Baily     5222
  16 William    Walter    5222
```

2. Using NOT EXISTS

List all customers' first name and the last name excluded customers' billing status is bad, pickup location is 5222 and booking date is on after Oct 5th, 2018.

```
SELECT DISTINCT c.FirstName, c.LastName
FROM RESERVATION r, CUSTOMER c
WHERE NOT EXISTS
(Select  cu.FIRSTNAME, cu.LASTNAME
FROM BILLING b, MAKES ma, CUSTOMER cu, RESERVATION rv, CREATES cr
WHERE ma.RESERVATIONID = rv.RESERVATIONID
AND rv.RESERVATIONID = cr.RESERVATIONID
AND cr.BILLINGID = b.BILLINGID
AND ma.LICENSENUMBER = cu.LICENSENUMBER
AND b.BILLINGSTATUS = 'BAD'
AND rv.PICKUPLOCATION = '5222'
AND ma.BOOKINGDATE >= '18-10-05');
```

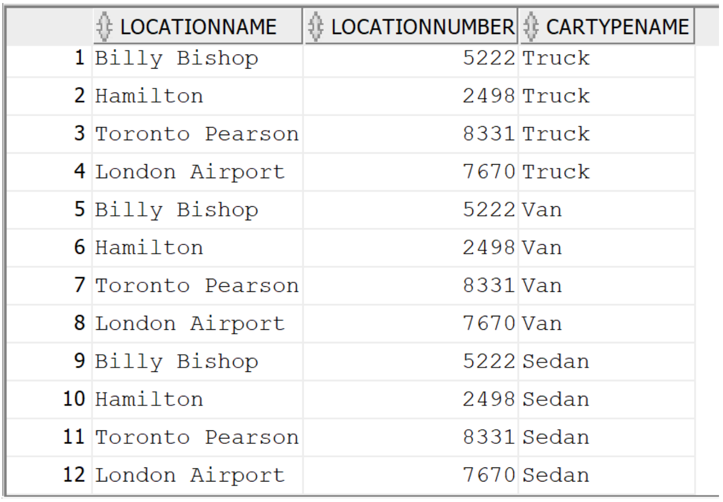


	FIRSTNAME	LASTNAME
1	Tina	Baily
2	Josephine	Mast
3	Paul	Arroyo
4	Katherine	Sisson
5	Casey	Trejo
6	Javid	Decster
7	William	Walter
8	Mattie	Warnock

3. Using UNION

List of Location name, Location Number, and Car_Type name excluded Location Number is ‘2331’ or Car_type is ‘Bus’.

```
SELECT DISTINCT lo1.LOCATIONNAME,lo1.LOCATIONNUMBER, ct1.CARTYPENAME
FROM LOCATION lo1, CAR_TYPE ct1
WHERE  NOT EXISTS
(SELECT lo2.LOCATIONNAME
FROM LOCATION lo2
WHERE lo2.LOCATIONNUMBER = '2332'
UNION
SELECT ct2.CARTYPENAME
FROM CAR_TYPE ct2
WHERE ct2.CARTYPENAME = 'Bus');
```

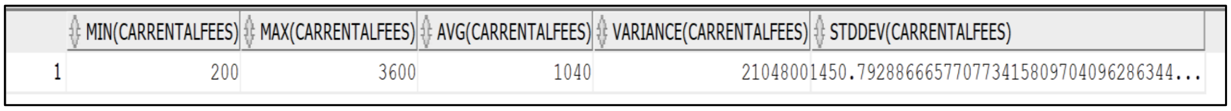


	LOCATIONNAME	LOCATIONNUMBER	CARTYPENAME
1	Billy Bishop	5222	Truck
2	Hamilton	2498	Truck
3	Toronto Pearson	8331	Truck
4	London Airport	7670	Truck
5	Billy Bishop	5222	Van
6	Hamilton	2498	Van
7	Toronto Pearson	8331	Van
8	London Airport	7670	Van
9	Billy Bishop	5222	Sedan
10	Hamilton	2498	Sedan
11	Toronto Pearson	8331	Sedan
12	London Airport	7670	Sedan

4. Using AGGREGATE FUNCTIONS

Find the minimum, maximum, average, variance, and standard deviation of the car rental cost of all customer

```
SELECT MIN(carrentalfees), MAX(carrentalfees), AVG(carrentalfees),
VARIANCE(carrentalfees), STDDEV(carrentalfees)
FROM BILLING;
```

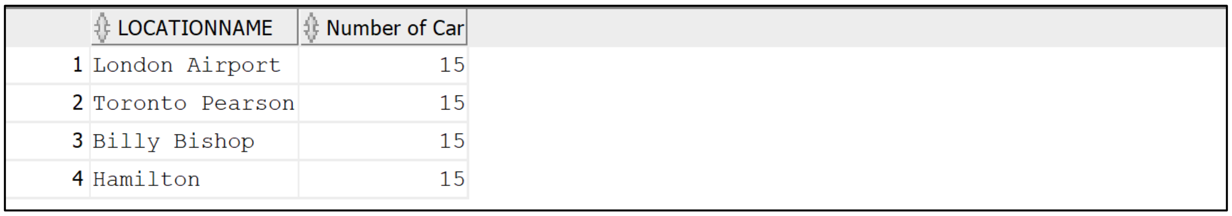


	MIN(CARRENTALFEES)	MAX(CARRENTALFEES)	AVG(CARRENTALFEES)	VARIANCE(CARRENTALFEES)	STDDEV(CARRENTALFEES)
1	200	3600	1040	2104800	1450.7928866657707734158097040962863 44...

5. Using GROUP

List how many car whose manufacturer is "Honda" there is depending on Location

```
SELECT location.LOCATIONNAME, count(*) as "Number of Car"
FROM Customer cus, car_model cm, Chosen ch, Makes ma, location, car
WHERE cus.LicenseNumber = ma.LicenseNumber
AND ma.ReservationId = ch.ReservationId
AND ch.RegistrationNumber = car.registrationnumber
AND cm.Manufacturer = 'Honda'
GROUP BY location.locationname;
```



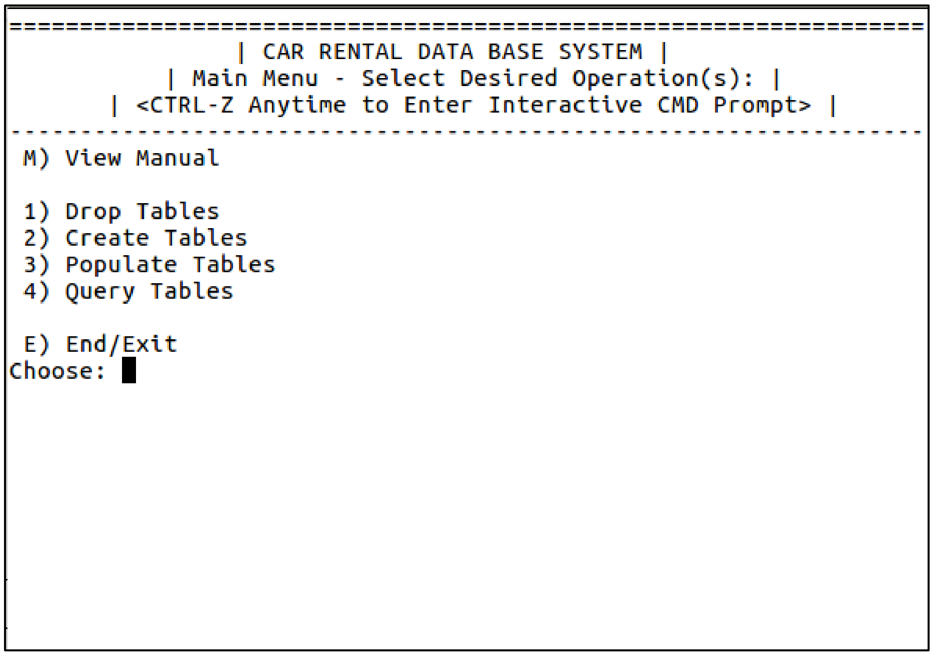
	LOCATIONNAME	Number of Car
1	London Airport	15
2	Toronto Pearson	15
3	Billy Bishop	15
4	Hamilton	15

UNIX Shell

[startOracle.sh]

```
#!/bin/sh

MainMenu()
{
        while [ true ]
        do
                clear

                echo
"==============================================================="
                echo "                | CAR RENTAL DATABASE SYSTEM |"
                echo "          | Main Menu - Select Desired Operation(s): |"
                echo "      | <CTRL-Z Anytime to Enter Interactive CMD Prompt> |"
                echo "---------------------------------------------------------------"
                echo " M) View Manual"
                echo " "
                echo " 1) Drop Tables"
                echo " 2) Create Tables"
                echo " 3) Populate Tables"
                echo " 4) Query Tables"
                echo " "
                echo " E) End/Exit"
                echo -n "Choose: "

                read CHOICE

                if [ ${CHOICE} -eq "0" ]
                then
                        echo "Nothing Here"
                elif [ ${CHOICE} -eq "1" ]
                then
                        bash drop_tables.sh
                        Pause
                elif [ ${CHOICE} -eq "2" ]
                then
                        bash create_tables.sh
                        Pause
                elif [ ${CHOICE} -eq "3" ]
                then
                        bash populate_tables.sh
                        Pause
                elif [ ${CHOICE} -eq "4" ]
                then
                        bash queries.sh
```

```
                    Pause
            else [ ${CHOICE} = "E" -o ${CHOICE} = "e" ]
                    exit 0
            fi
    done
}

#--COMMENTS BLOCK--
# Main Program
#--COMMENTS BLOCK--
ProgramStart()
{
    #StartMessage
    while [ 1 ]
    do
            MainMenu
    done
}

ProgramStart
exit
```



```
==================================================================
                    | CAR RENTAL DATA BASE SYSTEM |
               | Main Menu - Select Desired Operation(s): |
           | <CTRL-Z Anytime to Enter Interactive CMD Prompt> |
------------------------------------------------------------------
 M) View Manual

 1) Drop Tables
 2) Create Tables
 3) Populate Tables
 4) Query Tables

 E) End/Exit
Choose:
```

[drop_tables.sh]

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib

username="rsai"
password="oracleLABwork!"

sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" < ./dropTablesQueries.txt
```

[create_tables.sh]

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib

username="rsai"
password="oracleLABwork!"

sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" < ./createTablesQueries.txt
```

[Populate_tables.sh]

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib

username="rsai"
password="oracleLABwork!"

sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" < ./populateTablesQueries.txt
```

[queries.sh]

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib

username="rsai"
password="oracleLABwork!"

echo '1. List of Cars With More Than 7 Seats'
echo '2. List of Promotions That Are Less Than Or Equal To 70'
echo '3. List of Locations With LocationNumber Over 5000'
echo '4. List of Promotions That Are Percentages'
echo '5. List of Promotions That Are Amounts'

read input

if [ "$input" = "1" ]; then
   sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"  < ./query1.txt
elif [ "$input" = "2" ]; then
   sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"  < ./query2.txt
elif [ "$input" = "3" ]; then
   sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"  < ./query3.txt
elif [ "$input" = "4" ]; then
   sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"  < ./query4.txt
elif [ "$input" = "5" ]; then
   sqlplus64
"$username/$password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.
ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"  < ./query5.txt
else
   echo 'Invalid Input'
fi
```

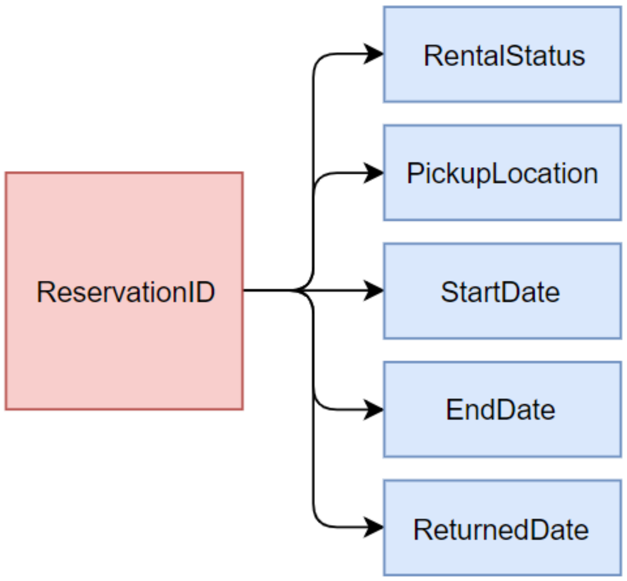
<u>Database Normalization</u>

ENTITY TABLES

1. Reservation

RESERVATION (<u>ReservationID</u>, RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate)

Functional Dependency:	ReservationID → RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate

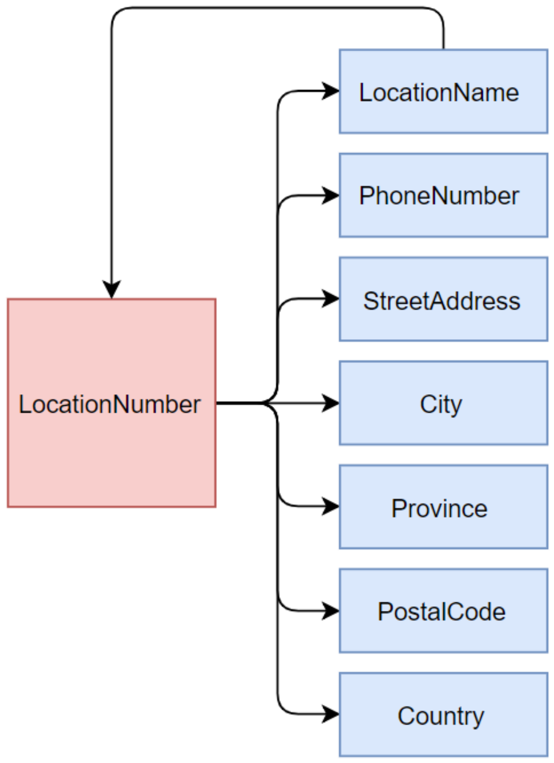


- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
  - In other words, the closure of ReservationID can get all other attributes and so ReservationID is a candidate key. Thus, this table is BCNF.

2. Location

LOCATION (<u>LocationNumber</u>, LocationName, PhoneNumber, StreetAddress, City, Province, PostalCode, Country)

Functional Dependency:	LocationNumber → LocationName, PhoneNumber, StreetAddress, City, Province, PostalCode, Country. LocationName → LocationNumber

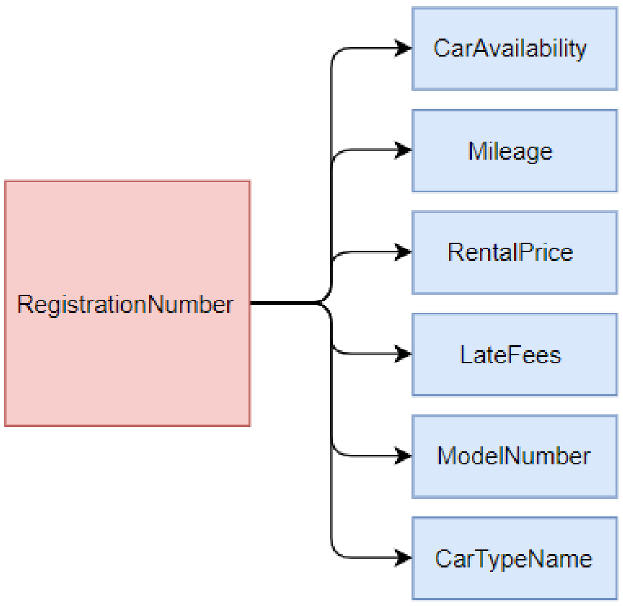


- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
  - LocationNumber is dependent on LocationName, but since LocationNumber is not a non-candidate key attribute, 3NF still holds.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
  - In other words, the closure of LocationNumber can get all other attributes and so LocationNumber is a candidate key. Thus, this table is BCNF.
  - Additionally, for the 2nd functional dependency, LocationName is a candidate key so BCNF still holds.

3. Car

CAR (RegistrationNumber, CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber, CarTypeName)

Functional Dependency:	RegistrationNumber → CarAvailiability, Mileage, RentalPrice, LateFees, ModelNumber, CarTypeName

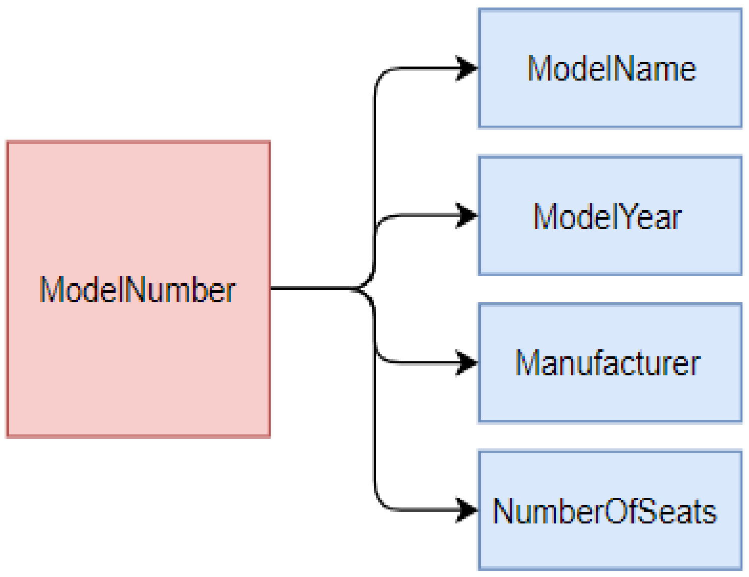


- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of RegistrationNumber can get all other attributes and so RegistrationNumber is the candidate key. Thus, this table is BCNF.

4. Car Model

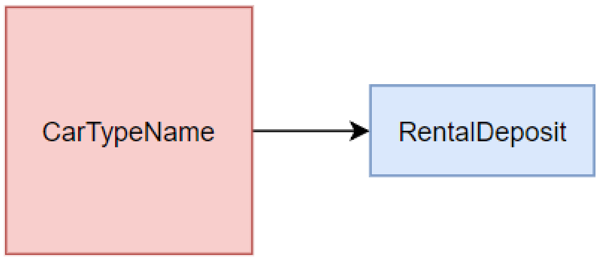
CAR_MODEL (<u>ModelNumber</u>, ModelName, ModelYear, Manufacturer, NumberOfSeats)

Functional Dependency:	ModelNumber → ModelName, ModelYear, Manufacturer, NumberOfSeats



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of ModelNumber can get all other attributes and so ModelNumber is the candidate key. Thus, this table is BCNF.

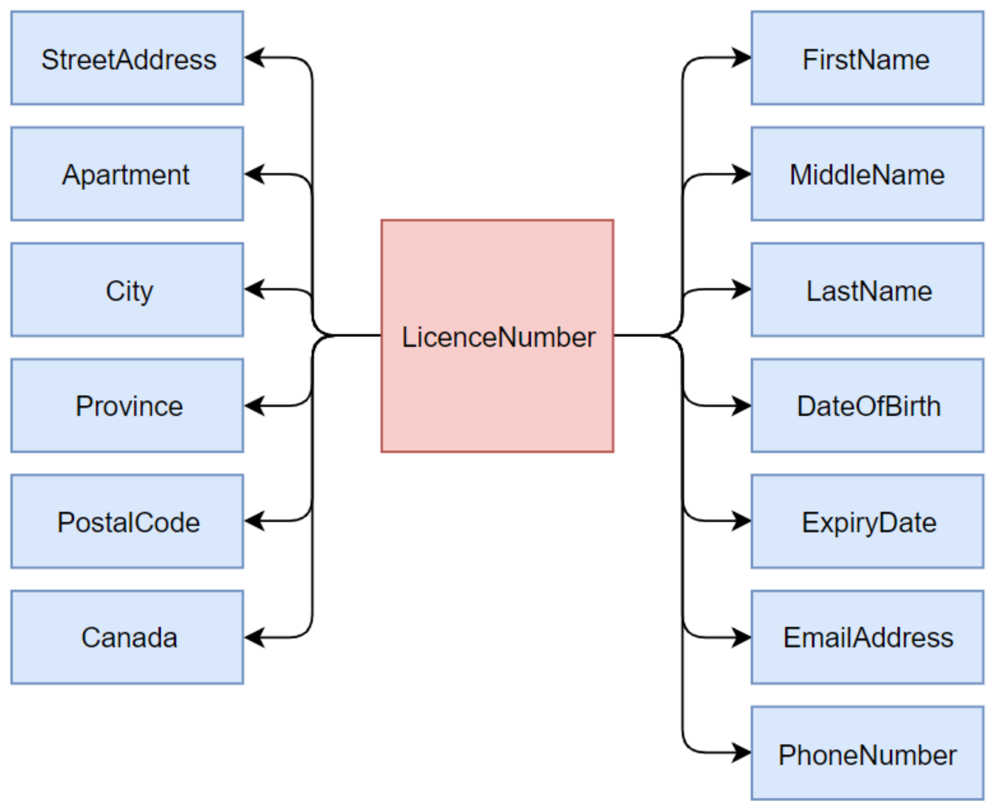
5. Car Type

CAR_TYPE (CarTypeName, RentalDeposit)	
Functional Dependency:	CarTypeName → RentalDeposit
	
• This table is 1NF because all values are atomic. • This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. • This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. • This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant. 　○ In other words, the closure of CarTypeName can get all other attributes and so CarTypeName is the candidate key. Thus, this table is BCNF.	

6. Customer

CUSTOMER (<u>LicenceNumber</u>, ExpiryDate, DateOfBirth, FirstName, MiddleName, LastName, EmailAddress, PhoneNumber, StreetAddress, Apartment, City, Province, PostalCode, Country)

Functional Dependency:	LicenceNumber → ExpiryDate, DateOfBirth, FirstName, MiddleName, LastName, Age, StreetAddress, Apartment, City, Province, PostalCode, Country, EmailAddress, PhoneNumber

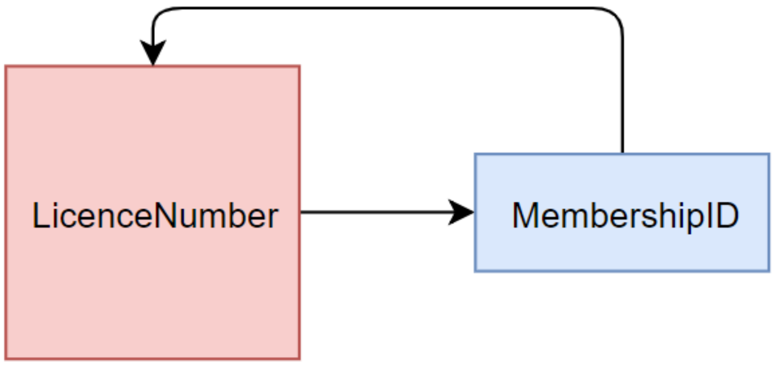


- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of LicenceNumber can get all other attributes and so LicenceNumber is a candidate key. Thus, this table is BCNF.

7. Member

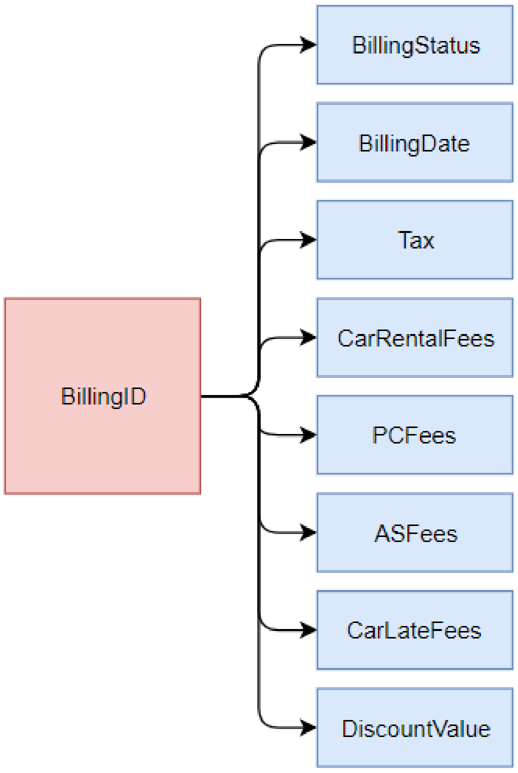
MEMBER (<u>LicenceNumber</u>, MembershipID)

Functional Dependency:	LicenceNumber → MembershipID MembershipID → LicenceNumber



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
    - LicenceNumber is dependent on MembershipID, but since LicenceNumber is not a non-candidate key attribute, 3NF still holds.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of LicenceNumber can get all other attributes and so LicenceNumber is a candidate key. Thus, this table is BCNF.
    - Additionally, for the 2nd functional dependency, MembershipID is a candidate key so BCNF still holds.

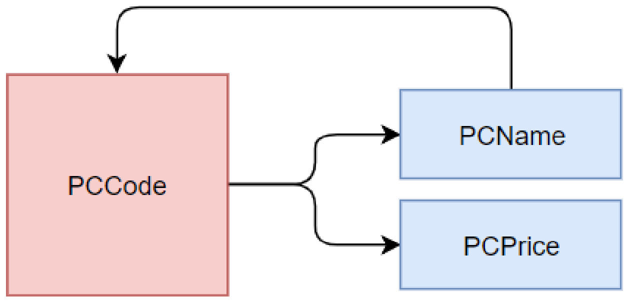
8. Billing

BILLING (<u>BillingID</u>, BillingStatus, BillingDate, Tax, CarRentalFees, PcFees, AsFees, CarLateFees, DiscountValue)	
Functional Dependency:	BillingID → BillingStatus, BillingDate, Tax, CarRentalFees, PCFees, ASFees, CarLateFees, DiscountValue
 BillingID BillingStatus BillingDate Tax CarRentalFees PCFees ASFees CarLateFees DiscountValue	
● This table is 1NF because all values are atomic. ● This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. ● This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. ● This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant. ○ In other words, the closure of BillingID can get all other attributes and so BillingID is a candidate key. Thus, this table is BCNF.	

9. Protections & Coverages

PROTECTIONS_COVERAGE (PCCode, PCName, PCPrice)

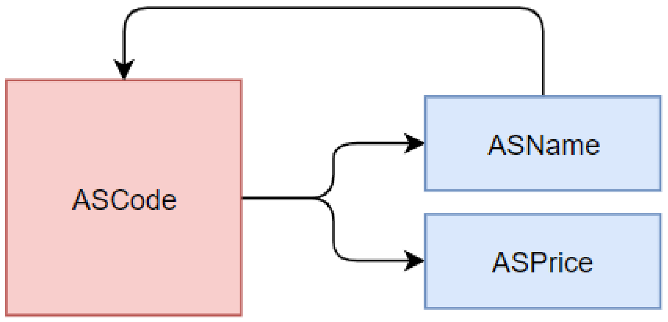
Functional Dependency:	PCCode → PCName, PCPrice PCName → PCCode



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
    - PCCode is dependent on PCName, but since PCCode is not a non-candidate key attribute, 3NF still holds.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of PCCode can get all other attributes and so PCCode is a candidate key. Thus, this table is BCNF.
    - Additionally, for the 2nd functional dependency, PCName is a candidate key so BCNF
      still holds.

10. Accessories & Services

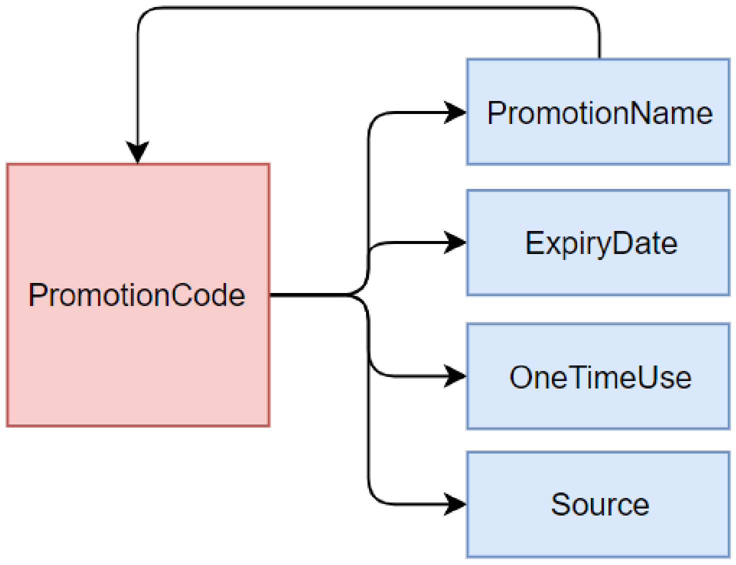
ACCESSORIES_SERVICES (<u>ASCode</u>, ASName, ASPrice)	
Functional Dependency:	ASCode → ASName, ASPrice ASName → ASCode



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
    - ASCode is dependent on ASName, but since ASCode is not a non-candidate key attribute, 3NF still holds.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of ASCode can get all other attributes and so ASCode is a candidate key. Thus, this table is BCNF.
    - Additionally, for the 2nd functional dependency, ASName is a candidate key so BCNF
      still holds.

11. Promotion

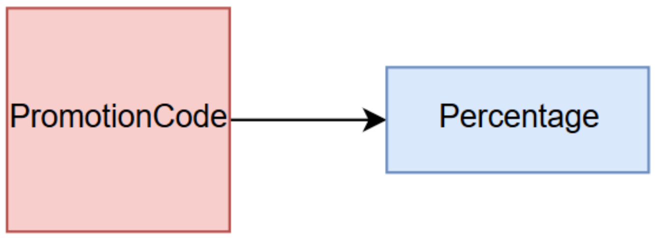
PROMOTION (<u>PromotionCode</u>, PromotionName, ExpiryDate, OneTimeUse, Source)	
Functional Dependency:	PromotionCode → PromotionName, ExpiryDate, OneTimeUse, Source PromotionName → PromotionCode



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
    - PromotionCode is dependent on PromotionName, but since PromotionCode is not a non-candidate key attribute, 3NF still holds.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of PromotionCode can get all other attributes and so PromotionCode is a candidate key. Thus, this table is BCNF.
    - Additionally, for the 2nd functional dependency, PromotionName is a candidate key so BCNF still holds.

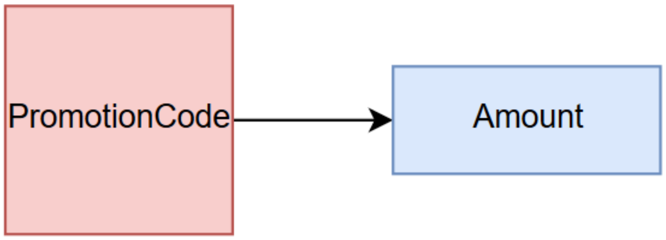
12. Promotion: Percentages

PROMOTION_PERCENTAGE (PromotionCode, Percentage)	
Functional Dependency:	PromotionCode → Percentage



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.
    - In other words, the closure of PromotionCode can get all other attributes and so PromotionCode is a candidate key. Thus, this table is BCNF.

13. Promotion: Amounts

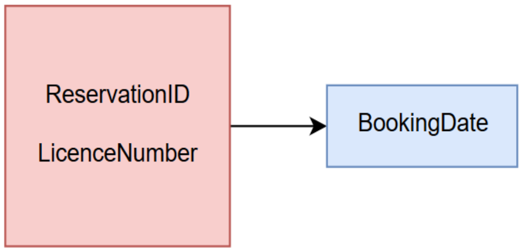
PROMOTION_AMOUNT (<u>PromotionCode</u>, Amount)	
Functional Dependency:	PromotionCode → Amount
	
• This table is 1NF because all values are atomic. • This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. • This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. • This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant. ○ In other words, the closure of PromotionCode can get all other attributes and so PromotionCode is a candidate key. Thus, this table is BCNF.	

RELATIONSHIP TABLES

1. Makes

MAKES (LicenceNumber, ReservationID, BookingDate)

Functional Dependency:	ReservationID → LicenceNumber
	ReservationID, LicenceNumber → BookingDate



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.

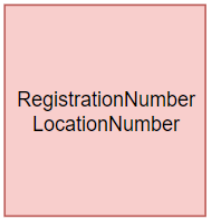
2. Chosen

CHOSEN (ReservationID, RegistrationNumber)

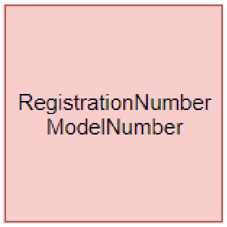
Functional Dependency:	ReservationID → RegistrationNumber

- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. In this case, there are no non-key attributes for the relationship.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. In this case, there are no non-key attributes for the relationship.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.

3. Has

HAS (<u>RegistrationNumber</u>, LocationNumber)	
Functional Dependency:	RegistrationNumber → LocationNumber
	
● This table is 1NF because all values are atomic. ● This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. In this case, there are no non-key attributes. ● This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. In this case, there are no non-key attributes. ● This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.	

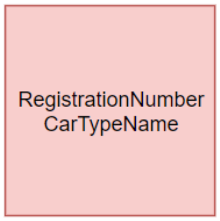
4. Contains

CONTAINS (<u>RegistrationNumber</u>, ModelNumber)	
Functional Dependency:	RegistrationNumber → ModelNumber
	
● This table is 1NF because all values are atomic. ● This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. In this case, there are no non-key attributes for the relationship. ● This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. In this case, there are no non-key attributes for the relationship. ● This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.	

5. Belongs

BELONGS (RegistrationNumber, CarTypeName)

Functional Dependency:	RegistrationNumber → CarTypeName



- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. In this case, there are no non-key attributes for the relationship.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. In this case, there are no non-key attributes for the relationship.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.

6. Includes

INCLUDES (ReservationID, PCCode, ASCode, PromotionCode)

Functional Dependency:	• ReservationID → PromotionCode • Since the mapping is N:M for the RESERVATION to PROTECTIONS_COVERAGES and RESERVATION to ACCESSORIES_SERVICES, they have no functional dependencies.

ReservationID
PromotionCode

- This table is 1NF because all values are atomic.
- This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. In this case, there are no non-key attributes for the relationship.
- This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. In this case, there are no non-key attributes for the relationship.
- This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.

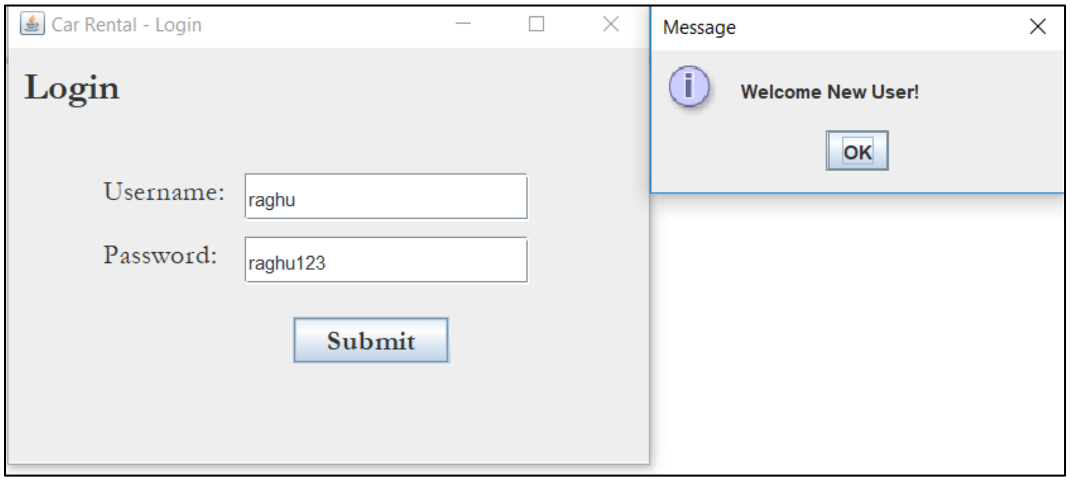
7. Creates

CREATES (<u>ReservationID</u>, BillingID)	
Functional Dependency:	ReservationID → BillingID
● This table is 1NF because all values are atomic. ● This table is 2NF because all non-key attributes are fully functionally dependent on the primary key. In this case, there are no non-key attributes for the relationship. ● This table is 3NF because all non-key attributes are non-transitively dependent on the primary key. In this case, there are no non-key attributes for the relationship. ● This table is BCNF because every nontrivial, left irreducible FD has a candidate key as its determinant.	

GUI

1. Screenshot Of Application

[Login Panel]

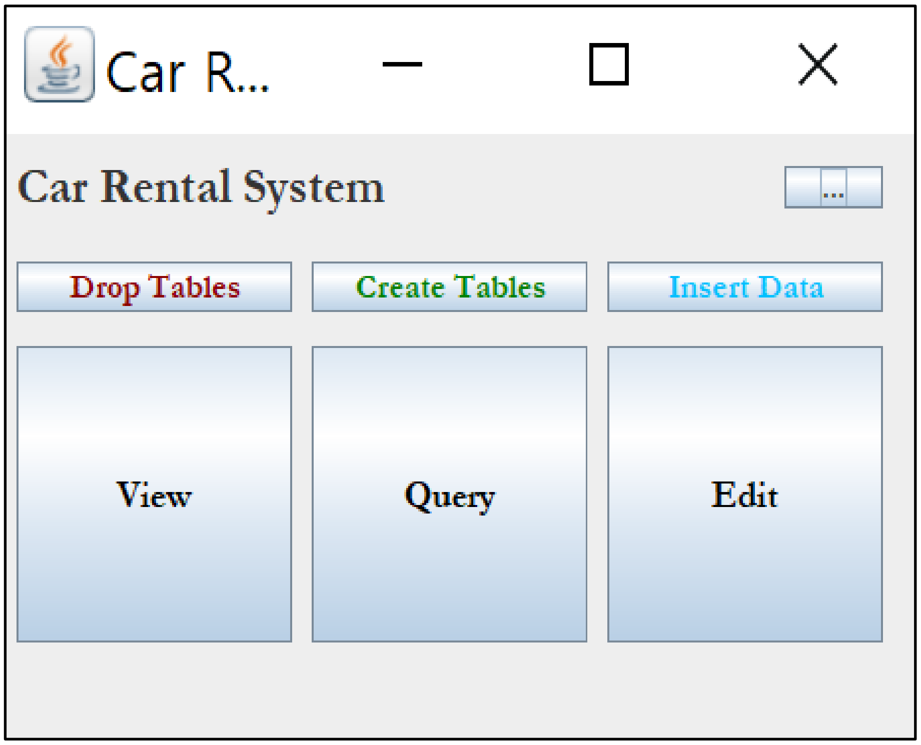


raghu

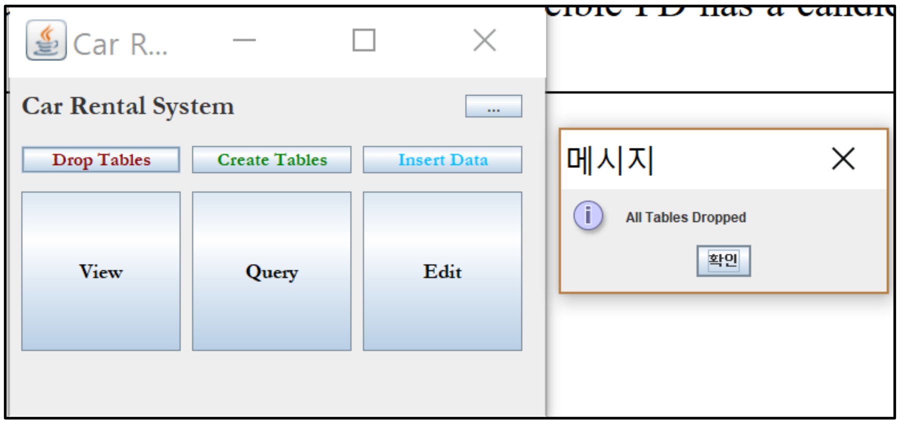
raghu123

Welcome New User!

[Main Panel]

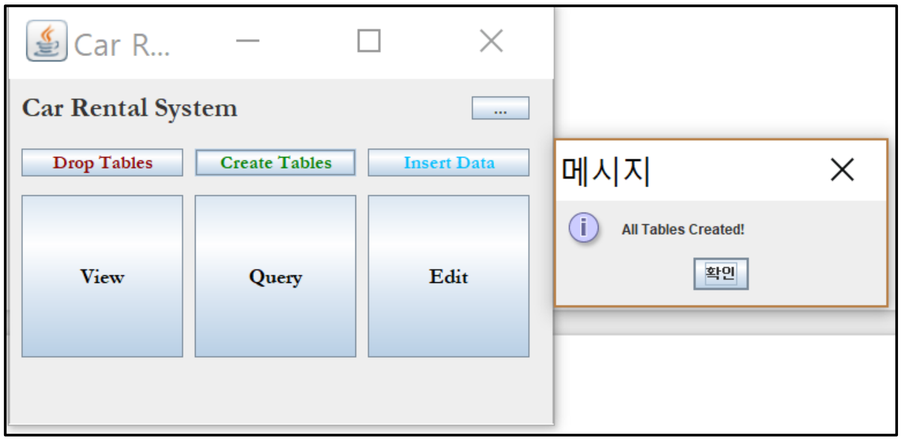


[Main Panel: Drop Tables]



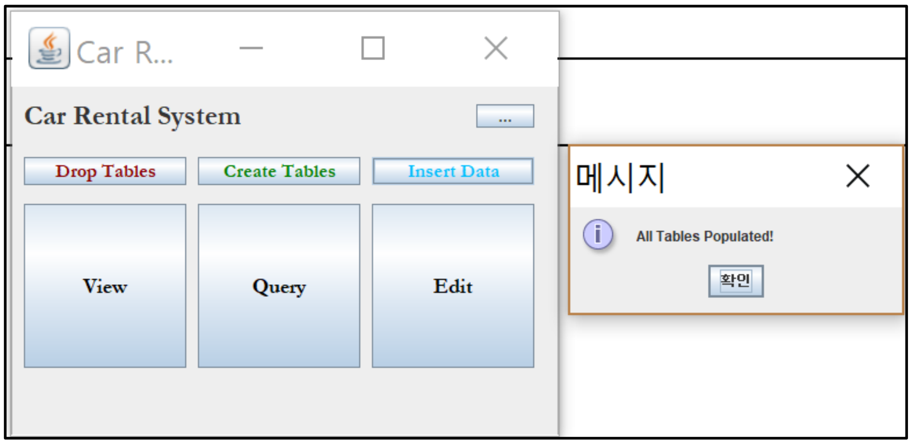
All Tables Dropped

[Main Panel: Create Tables]



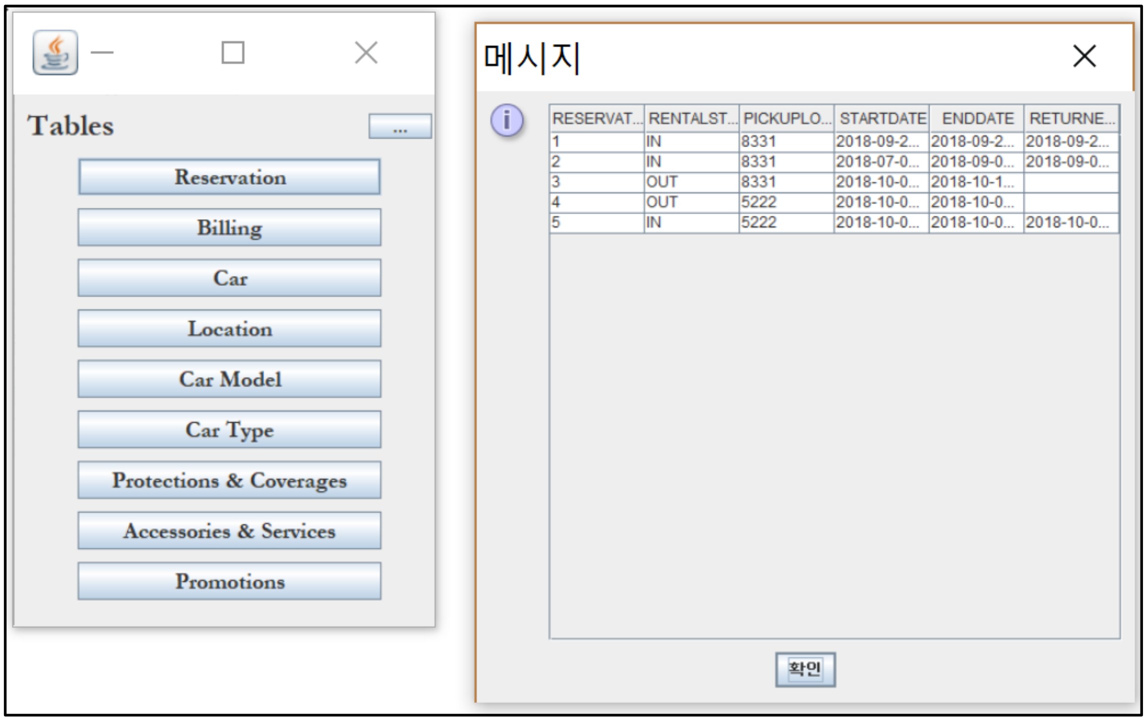
All Tables Created!

[Main Panel: Populate Tables]



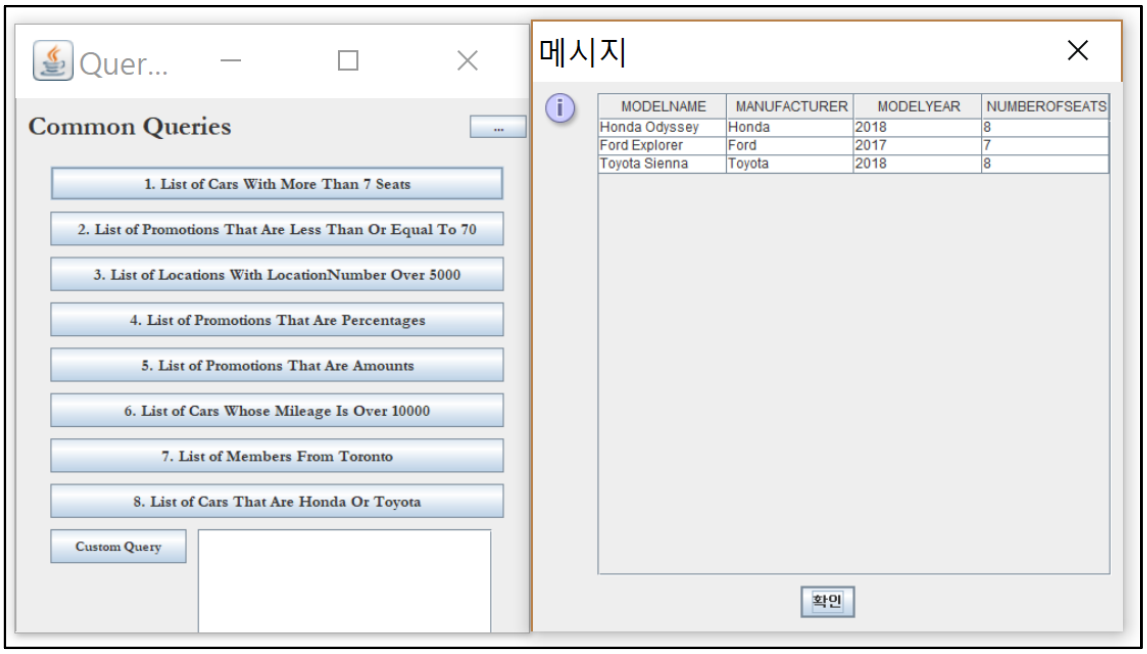
All Tables Populated!

[View Tables Panel: Reservation Table]



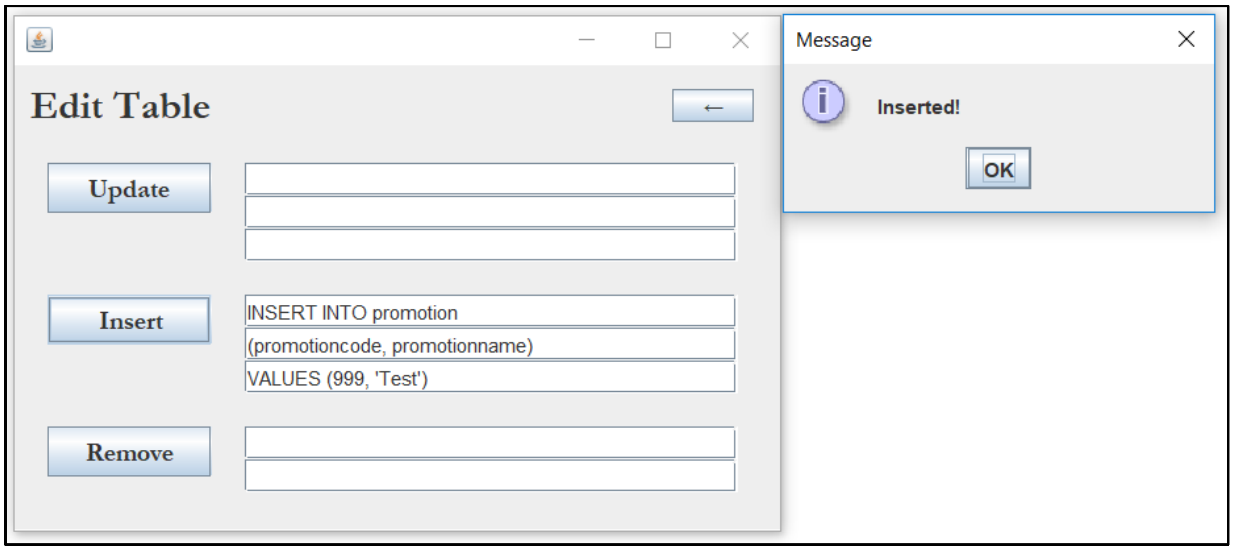
RESERVAT...	RENTALST...	PICKUPLO...	STARTDATE	ENDDATE	RETURNE...
1	IN	8331	2018-09-2...	2018-09-2...	2018-09-2...
2	IN	8331	2018-07-0...	2018-09-0...	2018-09-0...
3	OUT	8331	2018-10-0...	2018-10-1...	
4	OUT	5222	2018-10-0...	2018-10-0...	
5	IN	5222	2018-10-0...	2018-10-0...	2018-10-0...

[Common Queries Panel: List of cars with more than 7 seats.]


Quer...
Common Queries
...
1. List of Cars With More Than 7 Seats
2. List of Promotions That Are Less Than Or Equal To 70
3. List of Locations With LocationNumber Over 5000
4. List of Promotions That Are Percentages
5. List of Promotions That Are Amounts
6. List of Cars Whose Mileage Is Over 10000
7. List of Members From Toronto
8. List of Cars That Are Honda Or Toyota
Custom Query
메시지
확인

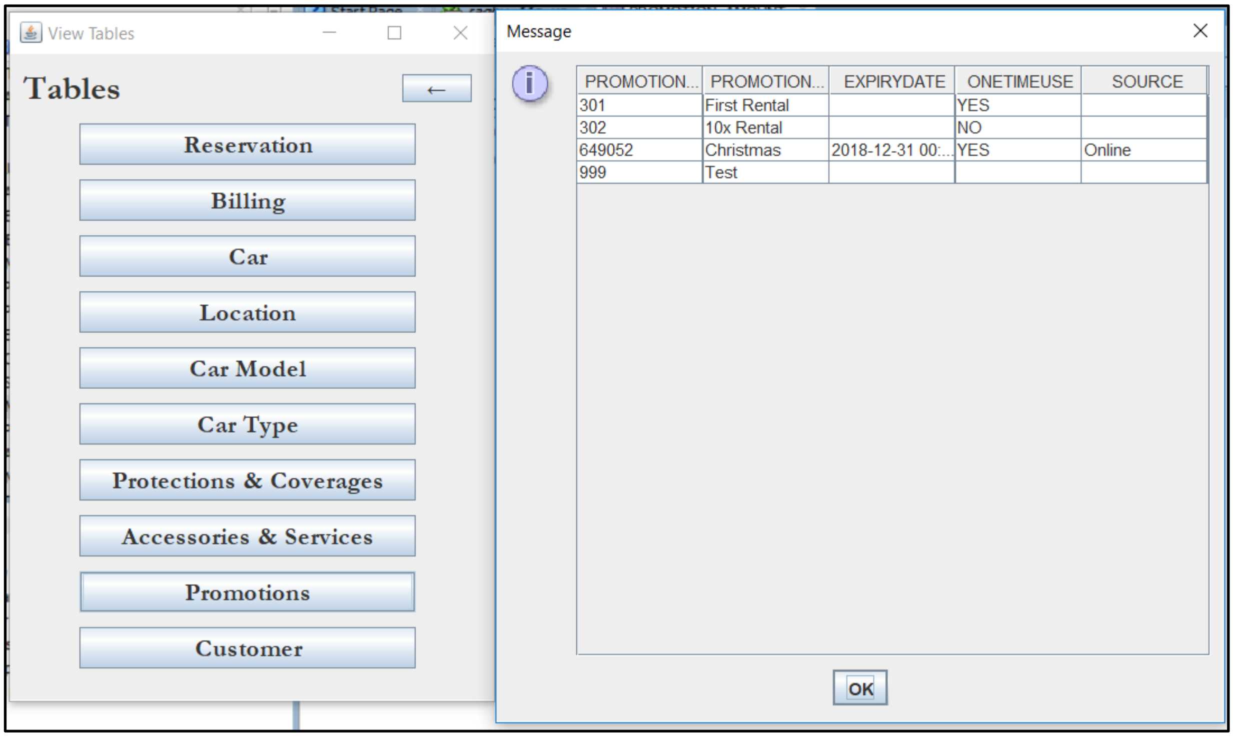
MODELNAME	MANUFACTURER	MODELYEAR	NUMBEROFSEATS
Honda Odyssey	Honda	2018	8
Ford Explorer	Ford	2017	7
Toyota Sienna	Toyota	2018	8

[Update Table Panel: Adding A New Promotion.]



```
INSERT INTO promotion
(promotioncode, promotionname)
VALUES (999, 'Test')
```

Inserted!



PROMOTION...	PROMOTION...	EXPIRYDATE	ONETIMEUSE	SOURCE
301	First Rental		YES	
302	10x Rental		NO	
649052	Christmas	2018-12-31 00:...	YES	Online
999	Test			

2. Source Code Of Application

DatabaseConnection.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DatabaseConnection {

   public Connection connection = null;

   public boolean createConnection() {
       try {
           Class.forName("oracle.jdbc.driver.OracleDriver");
       } catch (ClassNotFoundException e) {
           return false;
       }
       try {
           String dbURL1 = "jdbc:oracle:thin:raghu/raghu123@localhost:1521:xe";
           connection = DriverManager.getConnection(dbURL1);
           System.out.println("Connected!");
           return true;
       } catch (SQLException e) {
           e.printStackTrace();
           return false;
       }
   }

   public ResultSet executeQuery(String query) {
       try {
           Statement statement = connection.createStatement();
           ResultSet resultSet = statement.executeQuery(query);
           return resultSet;
       }
       catch (SQLException e) {
           e.printStackTrace();
           return null;
       }
   }
}
```

Login.java

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.*;
import java.util.HashMap;
import java.util.Map;

public class Login {

   private JFrame loginFrame;
   private JTextField usernameField;
   private JTextField passwordField;
   private JButton loginButton;
   private static Map<String, String> userCredentials;

   public static void main(String[] args) {
      loginRunnable();
   }

   public static void loginRunnable() {
      EventQueue.invokeLater(() -> {
         try {
            Login loginWindow = new Login();
            loginWindow.loginFrame.setVisible(true);
         }
         catch (Exception e) {
            e.printStackTrace();
         }
      });
   }

   public Login() {
      loginFrame = new JFrame();
      loginFrame.setTitle("Car Rental - Login");
      loginFrame.setBounds(100, 100, 420, 300);
      loginFrame.getContentPane().setLayout(null);
      loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JLabel carRentalLoginLabel = new JLabel("Login");
      carRentalLoginLabel.setFont(new Font("Garamond", Font.BOLD, 24));
      carRentalLoginLabel.setBounds(10, 0, 300, 50);
      loginFrame.getContentPane().add(carRentalLoginLabel);

      JLabel usernameLabel = new JLabel("Username:");
      usernameLabel.setFont(new Font("Garamond", Font.PLAIN, 18));
```

```
    usernameLabel.setBounds(60, 80, 80, 20);
    loginFrame.getContentPane().add(usernameLabel);

    JLabel passwordLabel = new JLabel("Password:");
    passwordLabel.setFont(new Font("Garamond", Font.PLAIN, 18));
    passwordLabel.setBounds(60, 120, 80, 20);
    loginFrame.getContentPane().add(passwordLabel);

    usernameField = new JTextField();
    usernameField.setBounds(150, 80, 180, 30);
    usernameField.setColumns(50);
    loginFrame.getContentPane().add(usernameField);

    passwordField = new JTextField();
    passwordField.setBounds(150, 120, 180, 30);
    passwordField.setColumns(50);
    loginFrame.getContentPane().add(passwordField);

    loginButton = new JButton("Submit");
    loginButton.setFont(new Font("Garamond", Font.BOLD, 18));
    loginButton.setBounds(180, 170, 100, 30);
    loginFrame.getContentPane().add(loginButton);

    userCredentials = new HashMap<>();

    loginButton.addActionListener(actionEvent -> {
        String username = usernameField.getText();
        String password = passwordField.getText();
        if (userCredentials.containsKey(username)) {
            if (userCredentials.get(username).equals(password)) {
                JOptionPane.showMessageDialog(loginFrame, "Success!");
                Menu menu = new Menu();
                menu.menuRunnable();
                loginFrame.setVisible(false);
            }
            else {
                JOptionPane.showMessageDialog(loginFrame, "Invalid Username or Password");
            }
        }
        else {
            userCredentials.put(username, password);
            JOptionPane.showMessageDialog(loginFrame, "Welcome New User!");
            Menu menu = new Menu();
            menu.menuRunnable();
            loginFrame.setVisible(false);
        }
    });
}}
```

Menu.java

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class Menu {

  private JFrame mainMenuFrame;
  DatabaseConnection databaseConnection = new DatabaseConnection();

  public static void main(String[] args) {
     menuRunnable();
  }

  public static void menuRunnable() {
     EventQueue.invokeLater(() -> {
        try {
           Menu menuWindow = new Menu();
           menuWindow.mainMenuFrame.setVisible(true);
        }
        catch (Exception e) {
           e.printStackTrace();
        }
     });
  }

  public Menu() {
     if (databaseConnection.createConnection()) {
        createMainMenuWindow();
     }
  }

  private void createMainMenuWindow() {
     mainMenuFrame = new JFrame();
     mainMenuFrame.setTitle("Car Rental System");
     mainMenuFrame.setBounds(100, 100, 480, 300);
     mainMenuFrame.getContentPane().setLayout(null);
     mainMenuFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

     JLabel carRentalSystemLabel = new JLabel("Car Rental System");
     carRentalSystemLabel.setFont(new Font("Garamond", Font.BOLD, 24));
```

```
        carRentalSystemLabel.setBounds(10, 0, 300, 50);
        mainMenuFrame.getContentPane().add(carRentalSystemLabel);

        JButton dropTablesButton = new JButton("Drop Tables");
        dropTablesButton.setForeground(new Color(139, 0, 0));
        dropTablesButton.setFont(new Font("Garamond", Font.BOLD, 16));
        dropTablesButton.setBounds(10, 60, 140, 24);
        mainMenuFrame.getContentPane().add(dropTablesButton);

        JButton createTablesButton = new JButton("Create Tables");
        createTablesButton.setForeground(new Color(0, 128, 0));
        createTablesButton.setFont(new Font("Garamond", Font.BOLD, 16));
        createTablesButton.setBounds(160, 60, 140, 24);
        mainMenuFrame.getContentPane().add(createTablesButton);

        JButton populateTablesButton = new JButton("Insert Data");
        populateTablesButton.setForeground(new Color(0, 191, 255));
        populateTablesButton.setFont(new Font("Garamond", Font.BOLD, 16));
        populateTablesButton.setBounds(310, 60, 140, 24);
        mainMenuFrame.getContentPane().add(populateTablesButton);

        DropTables.executeDropQueries(mainMenuFrame, databaseConnection,
dropTablesButton);
        CreateTables.executeCreateQueries(mainMenuFrame, databaseConnection,
createTablesButton);
        PopulateTables.executePopulateDataQueries(mainMenuFrame, databaseConnection,
populateTablesButton);

        JButton viewTablesButton = new JButton("View");
        viewTablesButton.setForeground(new Color(0, 0, 0));
        viewTablesButton.setFont(new Font("Garamond", Font.BOLD, 18));
        viewTablesButton.setBounds(10, 100, 140, 140);
        mainMenuFrame.getContentPane().add(viewTablesButton);

        JButton queryTablesButton = new JButton("Query");
        queryTablesButton.setForeground(new Color(0, 0, 0));
        queryTablesButton.setFont(new Font("Garamond", Font.BOLD, 18));
        queryTablesButton.setBounds(160, 100, 140, 140);
        mainMenuFrame.getContentPane().add(queryTablesButton);

        JButton updateTablesButton = new JButton("Edit");
        updateTablesButton.setForeground(new Color(0, 0, 0));
        updateTablesButton.setFont(new Font("Garamond", Font.BOLD, 18));
        updateTablesButton.setBounds(310, 100, 140, 140);
        mainMenuFrame.getContentPane().add(updateTablesButton);

        viewTablesButton.addActionListener(actionEvent -> {
            ViewTables viewTable = new ViewTables();
```

```
          viewTable.viewTableRunnable();
          mainMenuFrame.setVisible(false);
      });

      queryTablesButton.addActionListener(actionEvent -> {
          QueryTables queryTable = new QueryTables();
          queryTable.queryTableRunnable();
          mainMenuFrame.setVisible(false);
      });

      updateTablesButton.addActionListener(actionEvent -> {
          UpdateTables updateTable = new UpdateTables();
          updateTable.updateTablesRunnable();
          mainMenuFrame.setVisible(false);
      });

      JButton backButton = new JButton("←");
      backButton.setFont(new Font("Garamond", Font.BOLD, 14));
      backButton.setBounds(400, 15, 50, 20);
      mainMenuFrame.getContentPane().add(backButton);

      backButton.addActionListener(actionEvent -> {
          Login login = new Login();
          login.loginRunnable();
          mainMenuFrame.dispose();
      });
  }

  public static void executeButtonActionEvent(JButton tableButton, DatabaseConnection
databaseConnection, String query) {
      tableButton.addActionListener(actionEvent -> {
          try {
              ResultSet queryResult = databaseConnection.executeQuery(query);
              JTable queryResultTable = new JTable(buildTableModel(queryResult));
              JOptionPane.showMessageDialog(null, new JScrollPane(queryResultTable));
          }
          catch (SQLException e) {
              e.printStackTrace();
          }
      });
  }

  public static DefaultTableModel buildTableModel(ResultSet queryResult) throws
SQLException {

      ResultSetMetaData queryMetaData = queryResult.getMetaData();
      int columnCount = queryMetaData.getColumnCount();
      Vector<String> columnNames = new Vector<>();
```

```
        Vector<Vector<Object>> queryDataVector = new Vector<>();

        for (int columnNumber = 1; columnNumber <= columnCount; columnNumber++) {
            columnNames.add(queryMetaData.getColumnName(columnNumber));
        }

        while (queryResult.next()) {
            Vector<Object> tempDataVector = new Vector<>();
            for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
                tempDataVector.add(queryResult.getObject(columnIndex));
            }
            queryDataVector.add(tempDataVector);
        }
        return new DefaultTableModel(queryDataVector, columnNames);
    }
}
```

DropTables.java

```
import javax.swing.*;

public class DropTables {

    public static void executeDropQueries(JFrame mainMenuFrame, DatabaseConnection
databaseConnection, JButton dropTablesButton) {
        dropTablesButton.addActionListener(actionEvent -> {

            databaseConnection.executeQuery("DROP TABLE RESERVATION cascade
constraints purge");
            databaseConnection.executeQuery("DROP TABLE LOCATION cascade constraints
purge");
            databaseConnection.executeQuery("DROP TABLE CAR cascade constraints purge");
            databaseConnection.executeQuery("DROP TABLE CAR_MODEL cascade constraints
purge");
            databaseConnection.executeQuery("DROP TABLE CAR_TYPE cascade constraints
purge");
            databaseConnection.executeQuery("DROP TABLE CUSTOMER cascade constraints
purge");
            databaseConnection.executeQuery("DROP TABLE MEMBER cascade constraints
purge");
            databaseConnection.executeQuery("DROP TABLE BILLING cascade constraints
purge");
            databaseConnection.executeQuery("DROP TABLE PROTECTIONS_COVERAGES
cascade constraints purge");
            databaseConnection.executeQuery("DROP TABLE ACCESSORIES_SERVICES
cascade constraints purge");
            databaseConnection.executeQuery("DROP TABLE PROMOTION cascade constraints
purge");
```

```
        databaseConnection.executeQuery("DROP TABLE PROMOTION_PERCENTAGE
cascade constraints purge");
        databaseConnection.executeQuery("DROP TABLE PROMOTION_AMOUNT cascade
constraints purge");
        databaseConnection.executeQuery("DROP TABLE MAKES cascade constraints
purge");
        databaseConnection.executeQuery("DROP TABLE CHOSEN cascade constraints
purge");
        databaseConnection.executeQuery("DROP TABLE HAS cascade constraints purge");
        databaseConnection.executeQuery("DROP TABLE CONTAINS cascade constraints
purge");
        databaseConnection.executeQuery("DROP TABLE BELONGS cascade constraints
purge");
        databaseConnection.executeQuery("DROP TABLE INCLUDES cascade constraints
purge");
        databaseConnection.executeQuery("DROP TABLE CREATES cascade constraints
purge");
        databaseConnection.executeQuery("commit");

        JOptionPane.showMessageDialog(mainMenuFrame, "All Tables Dropped");
     });
  }
}
```

CreateTables.java

```
import javax.swing.*;

public class CreateTables {

  public static void executeCreateQueries (JFrame mainMenuFrame, DatabaseConnection
databaseConnection, JButton createTablesButton) {
     createTablesButton.addActionListener(actionEvent -> {

        databaseConnection.executeQuery("CREATE TABLE RESERVATION ( ReservationID
INTEGER PRIMARY KEY, RentalStatus VARCHAR2(3) CONSTRAINT
reservation_check_out_in CHECK (RentalStatus IN ('OUT', 'IN')), PickupLocation
VARCHAR2(18) NOT NULL, StartDate DATE NOT NULL, EndDate DATE NOT NULL,
ReturnedDate DATE )");
        databaseConnection.executeQuery("CREATE TABLE LOCATION ( LocationNumber
INTEGER PRIMARY KEY, LocationName VARCHAR2(18), PhoneNumber VARCHAR2(12),
StreetAddress VARCHAR2(50) NOT NULL, City VARCHAR2(20) NOT NULL, Province
VARCHAR2(2) DEFAULT 'ON', PostalCode VARCHAR2(7) NOT NULL, Country
VARCHAR2(20) DEFAULT 'Canada' )");
        databaseConnection.executeQuery("CREATE TABLE CAR ( RegistrationNumber
INTEGER PRIMARY KEY, CarAvailability VARCHAR2(3) CONSTRAINT car_check_out_in
CHECK (CarAvailability IN ('OUT', 'IN')), Mileage INTEGER, RentalPrice INTEGER NOT
NULL, LateFees INTEGER, ModelNumber INTEGER NOT NULL )");
```

```
        databaseConnection.executeQuery("CREATE TABLE CAR_MODEL ( ModelNumber
INTEGER PRIMARY KEY, ModelName VARCHAR2(18), ModelYear INTEGER, Manufacturer
VARCHAR2(18), NumberOfSeats INTEGER )");
        databaseConnection.executeQuery("CREATE TABLE CAR_TYPE ( CarTypeName
VARCHAR2(18) PRIMARY KEY, RentalDeposit INTEGER )");
        databaseConnection.executeQuery("CREATE TABLE CUSTOMER ( LicenseNumber
VARCHAR2(18) PRIMARY KEY, ExpiryDate DATE NOT NULL, DateOfBirth DATE NOT
NULL, FirstName VARCHAR2(20) NOT NULL, MiddleName VARCHAR2(1), LastName
VARCHAR2(30) NOT NULL, EmailAddress VARCHAR2(50), PhoneNumber VARCHAR2(12)
NOT NULL, StreetAddress VARCHAR2(50) NOT NULL, Apartment VARCHAR2(5), City
VARCHAR2(20) NOT NULL, Province VARCHAR2(2) DEFAULT 'ON', PostalCode
VARCHAR2(7) NOT NULL, Country VARCHAR2(20) DEFAULT 'Canada' )");
        databaseConnection.executeQuery("CREATE TABLE MEMBER ( LicenseNumber
VARCHAR2(18) REFERENCES CUSTOMER(LicenseNumber), MembershipID INTEGER )");
        databaseConnection.executeQuery("CREATE TABLE BILLING ( BillingID INTEGER
PRIMARY KEY, BillingStatus VARCHAR2(4) CONSTRAINT billing_status_check_good_bad
CHECK (BillingStatus IN ('GOOD', 'BAD')), BillingDate DATE NOT NULL, Tax INTEGER NOT
NULL, CarRentalFees INTEGER NOT NULL, PCFees INTEGER, ASFees INTEGER,
CarLateFees INTEGER, DiscountValue INTEGER )");
        databaseConnection.executeQuery("CREATE TABLE PROTECTIONS_COVERAGES
( PCCode INTEGER PRIMARY KEY, PCName VARCHAR2(30), PCPrice INTEGER )");
        databaseConnection.executeQuery("CREATE TABLE ACCESSORIES_SERVICES (
ASCode INTEGER PRIMARY KEY, ASName VARCHAR2(30), ASPrice INTEGER )");
        databaseConnection.executeQuery("CREATE TABLE PROMOTION ( PromotionCode
INTEGER PRIMARY KEY, PromotionName VARCHAR2(30), ExpiryDate DATE,
OneTimeUse VARCHAR2(3) CONSTRAINT one_time_use_check_yes_no CHECK
(OneTimeUse IN ('YES', 'NO')), Source VARCHAR2(50) )");
        databaseConnection.executeQuery("CREATE TABLE PROMOTION_PERCENTAGE (
PromotionCode INTEGER REFERENCES PROMOTION(PromotionCode), Percentage
INTEGER, PRIMARY KEY(PromotionCode) )");
        databaseConnection.executeQuery("CREATE TABLE PROMOTION_AMOUNT (
PromotionCode INTEGER REFERENCES PROMOTION(PromotionCode), Amount
INTEGER, PRIMARY KEY(PromotionCode) )");
        databaseConnection.executeQuery("CREATE TABLE MAKES ( LicenseNumber
VARCHAR2(18) REFERENCES CUSTOMER(LicenseNumber), ReservationID INTEGER
REFERENCES RESERVATION(ReservationID), BookingDate DATE, PRIMARY
KEY(LicenseNumber,ReservationID) )");
        databaseConnection.executeQuery("CREATE TABLE CHOSEN ( ReservationID
INTEGER REFERENCES RESERVATION(ReservationID), RegistrationNumber INTEGER
REFERENCES CAR(RegistrationNumber), PRIMARY KEY(ReservationID,
RegistrationNumber) )");
        databaseConnection.executeQuery("CREATE TABLE HAS ( LocationNumber
INTEGER REFERENCES LOCATION(LocationNumber), RegistrationNumber INTEGER
REFERENCES CAR(RegistrationNumber), PRIMARY KEY(LocationNumber,
RegistrationNumber) )");
        databaseConnection.executeQuery("CREATE TABLE CONTAINS (
RegistrationNumber INTEGER REFERENCES CAR(RegistrationNumber), ModelNumber
INTEGER REFERENCES CAR_MODEL(ModelNumber), PRIMARY
KEY(RegistrationNumber, ModelNumber) )");
        databaseConnection.executeQuery("CREATE TABLE BELONGS ( RegistrationNumber INTEGER REFERENCES CAR(RegistrationNumber), CarTypeName VARCHAR2(18) REFERENCES CAR_TYPE(CarTypeName), PRIMARY KEY(RegistrationNumber, CarTypeName) )");
        databaseConnection.executeQuery("CREATE TABLE INCLUDES ( ReservationID INTEGER REFERENCES RESERVATION(ReservationID), PCCode INTEGER REFERENCES PROTECTIONS_COVERAGES(PCCode), ASCode INTEGER REFERENCES ACCESSORIES_SERVICES(ASCode), PromotionCode INTEGER REFERENCES PROMOTION(PromotionCode), PRIMARY KEY(ReservationID, PCCode, ASCode, PromotionCode) )");
        databaseConnection.executeQuery("CREATE TABLE CREATES ( ReservationID INTEGER REFERENCES RESERVATION(ReservationID), BillingID INTEGER REFERENCES BILLING(BillingID), PRIMARY KEY(ReservationID, BillingID) )");

        JOptionPane.showMessageDialog(mainMenuFrame, "All Tables Created!");
    });
  }
}
```

PopulateTables.java

```
import javax.swing.*;

public class PopulateTables {

  public static void executePopulateDataQueries(JFrame mainMenuFrame, DatabaseConnection databaseConnection, JButton populateTablesButton) {
    populateTablesButton.addActionListener(actionEvent -> {

        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber, StreetAddress, City, PostalCode) VALUES ('D3101-69581-63862', '2020/07/22', '1974/12/04', 'Paul', 'Arroyo', 'PaulDArroyo@dayrep.com', '604-431-6663', '1037 Adelaide St', 'Toronto', 'M5H 1P6')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber, StreetAddress, City, PostalCode) VALUES ('D2345-34567-45678', '2022/09/12', '1993/01/10', 'Javid', 'Decster', 'javidD@gmail.com', '647-222-2222', '100 Church St', 'Toronto', 'M2M 3B2')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber, StreetAddress, City, PostalCode) VALUES ('D4020-88182-19796', '2021/03/28', '1966/03/15', 'Katherine', 'Sisson', 'KatherineDSisson@armyspy.com', '519-570-6264', '4332 Water Street', 'Kitchener','N2H 5A5')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber, ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber, StreetAddress, City, PostalCode) VALUES ('D1416-89395-92248', '2024/07/20', '1992/11/10',
'Casey', 'Trejo', 'CaseyRTrejo@jourrapide.com', '519-473-1266', '3469 Hyde Park Road',
'London', 'N6H 3S2')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber,
ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber,
StreetAddress, City, PostalCode) VALUES ('D6960-19171-26137', '2024/11/21', '1995/12/08',
'Josephine', 'Mast', 'JosephineNMast@teleworm.us', '519-293-2149', '2991 Baker Street',
'London', 'N0N 0N0')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber,
ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber,
StreetAddress, City, PostalCode) VALUES ('D4348-32119-11009', '2024/05/27', '1990/08/03',
'William', 'Walter', 'WilliamDWalter@teleworm.us', '905-621-4183', '3830 Toy Avenue', 'Ajax',
'L1S 6L6')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber,
ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber,
StreetAddress, City, PostalCode) VALUES ('D9133-32635-40061', '2022/09/02', '1976/02/26',
'Tina', 'Baily', 'TinaABaily@teleworm.us', '905-391-1159', '2605 Toy Avenue', 'Kitchener', 'L1W
3N9')");
        databaseConnection.executeQuery("INSERT INTO CUSTOMER (LicenseNumber,
ExpiryDate, DateOfBirth, FirstName, LastName, EmailAddress, PhoneNumber,
StreetAddress, City, PostalCode) VALUES ('D1601-90328-66848', '2024/09/12', '1994/10/30',
'Mattie', 'Warnock', 'MattieRWarnock@teleworm.us', '647-223-0149', '4093 Islington Ave',
'Toronto', 'M8V 3B6')");

        databaseConnection.executeQuery("INSERT INTO MEMBER (LicenseNumber,
MembershipID) VALUES ('D1601-90328-66848', 1001)");
        databaseConnection.executeQuery("INSERT INTO MEMBER (LicenseNumber,
MembershipID) VALUES ('D4348-32119-11009', 1002)");
        databaseConnection.executeQuery("INSERT INTO MEMBER (LicenseNumber,
MembershipID) VALUES ('D9133-32635-40061', 1003)");
        databaseConnection.executeQuery("INSERT INTO MEMBER (LicenseNumber,
MembershipID) VALUES ('D4020-88182-19796', 1004)");
        databaseConnection.executeQuery("INSERT INTO MEMBER (LicenseNumber,
MembershipID) VALUES ('D3101-69581-63862', 1005)");

        databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (716830987, 'IN',
148299, 120, 20, 18187118415)");
        databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (416341540, 'IN',
98213, 120, 20, 53517077183)");
        databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (448632952, 'IN',
119096, 120, 20, 14685153720)");
        databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (830323748, 'IN',
44556, 80, 20, 14040211286)");
        databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (624749069, 'IN',
54522, 80, 20, 23552234653)");
	databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (232940401, 'IN',
74632, 100, 20, 46568522408)");
	databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (407177022, 'IN',
67354, 100, 20, 14332249770)");
	databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (956503875, 'IN',
132452, 60, 10, 82381133464)");
	databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (259930739, 'IN',
23456, 100, 20, 17210491613)");
	databaseConnection.executeQuery("INSERT INTO CAR (RegistrationNumber,
CarAvailability, Mileage, RentalPrice, LateFees, ModelNumber) VALUES (257923645, 'IN',
98765, 100, 20, 12760346714)");

	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (18187118415, 'Ford
Explorer', 2017, 'Ford', 7)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (53517077183, 'Ford
F150', 2018, 'Ford', 5)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (14685153720, 'Jeep
Wrangler', 2016, 'Jeep', 4)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (14040211286, 'Toyota
Sienna', 2018, 'Toyota', 8)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (23552234653, 'Honda
Odyssey', 2018, 'Honda', 8)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (46568522408, 'Honda
Accord', 2018, 'Honda', 5)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (14332249770, 'Toyota
Camry', 2018, 'Toyota', 5)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (82381133464, 'Honda
Civic', 2017, 'Honda', 4)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (17210491613, 'Lexus
GS', 2018, 'Lexus', 5)");
	databaseConnection.executeQuery("INSERT INTO CAR_MODEL (ModelNumber,
ModelName, ModelYear, Manufacturer, NumberOfSeats) VALUES (12760346714, 'Toyota
Corolla', 2017, 'Toyota', 5)");
```

```
	databaseConnection.executeQuery("INSERT INTO CAR_TYPE (CarTypeName, 
RentalDeposit) VALUES ('Truck', 500)");
	databaseConnection.executeQuery("INSERT INTO CAR_TYPE (CarTypeName, 
RentalDeposit) VALUES ('Van', 400)");
	databaseConnection.executeQuery("INSERT INTO CAR_TYPE (CarTypeName, 
RentalDeposit) VALUES ('Sedan', 300)");

	databaseConnection.executeQuery("INSERT INTO LOCATION (LocationNumber, 
LocationName, PhoneNumber, StreetAddress, City, PostalCode) VALUES (5222, 'Billy 
Bishop', '416-777-1618', '161A Bay Street', 'Toronto', 'M5J 2S1')");
	databaseConnection.executeQuery("INSERT INTO LOCATION (LocationNumber, 
LocationName, PhoneNumber, StreetAddress, City, PostalCode) VALUES (2498, 'Hamilton', 
'905-525-1400', '237 Main Street East', 'Hamilton', 'L8N 1H4')");
	databaseConnection.executeQuery("INSERT INTO LOCATION (LocationNumber, 
LocationName, PhoneNumber, StreetAddress, City, PostalCode) VALUES (8331, 'Toronto 
Pearson', '905-676-1100', '5990 Airport Road', 'Toronto', 'L5P 1B2')");
	databaseConnection.executeQuery("INSERT INTO LOCATION (LocationNumber, 
LocationName, PhoneNumber, StreetAddress, City, PostalCode) VALUES (7670, 'London 
Airport', '519-451-8400', '1750 Crumlin Road', 'London', 'N5V 3B6')");

	databaseConnection.executeQuery("INSERT INTO RESERVATION (ReservationID, 
RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate) VALUES (1, 'IN', '8331', 
'2018/09/28', '2018/09/29', '2018/09/29')");
	databaseConnection.executeQuery("INSERT INTO RESERVATION (ReservationID, 
RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate) VALUES (2, 'IN', '8331', 
'2018/07/02', '2018/09/06', '2018/09/06')");
	databaseConnection.executeQuery("INSERT INTO RESERVATION (ReservationID, 
RentalStatus, PickupLocation, StartDate, EndDate) VALUES (3, 'OUT', '8331', '2018/10/01', 
'2018/10/10')");
	databaseConnection.executeQuery("INSERT INTO RESERVATION (ReservationID, 
RentalStatus, PickupLocation, StartDate, EndDate) VALUES (4, 'OUT', '5222', '2018/10/04', 
'2018/10/06')");
	databaseConnection.executeQuery("INSERT INTO RESERVATION (ReservationID, 
RentalStatus, PickupLocation, StartDate, EndDate, ReturnedDate) VALUES (5, 'IN', '5222', 
'2018/10/03', '2018/10/03', '2018/10/04')");

	databaseConnection.executeQuery("INSERT INTO PROTECTIONS_COVERAGES 
(PCCode, PCName, PCPrice) VALUES (101, 'Loss Damage Waiver', 40)");
	databaseConnection.executeQuery("INSERT INTO PROTECTIONS_COVERAGES 
(PCCode, PCName, PCPrice) VALUES (102, 'Personal Accident Insurance', 10)");
	databaseConnection.executeQuery("INSERT INTO PROTECTIONS_COVERAGES 
(PCCode, PCName, PCPrice) VALUES (103, 'Personal Effects Protection', 5)");

	databaseConnection.executeQuery("INSERT INTO ACCESSORIES_SERVICES 
(ASCode, ASName, ASPrice) VALUES (201, 'Additional Driver', 10)");
	databaseConnection.executeQuery("INSERT INTO ACCESSORIES_SERVICES 
(ASCode, ASName, ASPrice) VALUES (202, 'Travel Tab', 25)");
	databaseConnection.executeQuery("INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice) VALUES (203, 'GPS', 10)");
        databaseConnection.executeQuery("INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice) VALUES (204, 'Extended Roadside Assistance', 10)");
        databaseConnection.executeQuery("INSERT INTO ACCESSORIES_SERVICES
(ASCode, ASName, ASPrice) VALUES (205, 'Child Safety Seat', 5)");

        databaseConnection.executeQuery("INSERT INTO PROMOTION (PromotionCode,
PromotionName, OneTimeUse) VALUES (301, 'First Rental', 'YES')");
        databaseConnection.executeQuery("INSERT INTO PROMOTION (PromotionCode,
PromotionName, OneTimeUse) VALUES (302, '10x Rental', 'NO')");
        databaseConnection.executeQuery("INSERT INTO PROMOTION (PromotionCode,
PromotionName, ExpiryDate, OneTimeUse, Source) VALUES (649052, 'Christmas',
'2018/12/31', 'YES', 'Online')");

        databaseConnection.executeQuery("INSERT INTO PROMOTION_PERCENTAGE
(PromotionCode, Percentage) VALUES (649052, 10)");

        databaseConnection.executeQuery("INSERT INTO PROMOTION_AMOUNT
(PromotionCode, Amount) VALUES (301, 100)");
        databaseConnection.executeQuery("INSERT INTO PROMOTION_AMOUNT
(PromotionCode, Amount) VALUES (302, 50)");

        databaseConnection.executeQuery("INSERT INTO BILLING (BillingID, BillingStatus,
BillingDate, Tax, CarRentalFees, PCFees, ASFees, DiscountValue) VALUES (1, 'GOOD',
'2018/09/29', 13, 200, 80, 10, 100)");
        databaseConnection.executeQuery("INSERT INTO BILLING (BillingID, BillingStatus,
BillingDate, Tax, CarRentalFees) VALUES (2, 'GOOD', '2018/09/06', 13, 3600)");
        databaseConnection.executeQuery("INSERT INTO BILLING (BillingID, BillingStatus,
BillingDate, Tax, CarRentalFees, PCFees, ASFees, DiscountValue) VALUES (3, 'BAD',
'2018/10/01', 13, 800, 400, 100, 10)");
        databaseConnection.executeQuery("INSERT INTO BILLING (BillingID, BillingStatus,
BillingDate, Tax, CarRentalFees, PCFees, ASFees) VALUES (4, 'BAD', '2018/10/04', 13, 360,
30, 30)");
        databaseConnection.executeQuery("INSERT INTO BILLING (BillingID, BillingStatus,
BillingDate, Tax, CarRentalFees, CarLateFees) VALUES (5, 'GOOD', '2018/10/04', 13, 240,
20)");

        databaseConnection.executeQuery("INSERT INTO MAKES (LicenseNumber,
ReservationID, BookingDate) VALUES ('D3101-69581-63862', 1, '2018/09/27')");
        databaseConnection.executeQuery("INSERT INTO MAKES (LicenseNumber,
ReservationID, BookingDate) VALUES ('D2345-34567-45678', 2, '2018/07/01')");
        databaseConnection.executeQuery("INSERT INTO MAKES (LicenseNumber,
ReservationID, BookingDate) VALUES ('D4020-88182-19796', 3, '2018/09/28')");
        databaseConnection.executeQuery("INSERT INTO MAKES (LicenseNumber,
ReservationID, BookingDate) VALUES ('D1416-89395-92248', 4, '2018/10/03')");
        databaseConnection.executeQuery("INSERT INTO MAKES (LicenseNumber,
ReservationID, BookingDate) VALUES ('D6960-19171-26137', 5, '2018/10/02')");
```

```
        databaseConnection.executeQuery("INSERT INTO CHOSEN (ReservationID,
RegistrationNumber) VALUES (1, 716830987)");
        databaseConnection.executeQuery("INSERT INTO CHOSEN (ReservationID,
RegistrationNumber) VALUES (2, 956503875)");
        databaseConnection.executeQuery("INSERT INTO CHOSEN (ReservationID,
RegistrationNumber) VALUES (3, 407177022)");
        databaseConnection.executeQuery("INSERT INTO CHOSEN (ReservationID,
RegistrationNumber) VALUES (4, 232940401)");
        databaseConnection.executeQuery("INSERT INTO CHOSEN (ReservationID,
RegistrationNumber) VALUES (5, 448632952)");

        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (5222, 716830987)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (2498, 416341540)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (8331, 448632952)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (7670, 830323748)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (5222, 624749069)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (2498, 232940401)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (8331, 407177022)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (7670, 956503875)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (7670, 259930739)");
        databaseConnection.executeQuery("INSERT INTO HAS (LocationNumber,
RegistrationNumber) VALUES (8331, 257923645)");

        databaseConnection.executeQuery("INSERT INTO INCLUDES (ReservationID,
PCCode, ASCode, PromotionCode) VALUES (1, 101, 205, 301)");
        databaseConnection.executeQuery("INSERT INTO INCLUDES (ReservationID,
PCCode, ASCode, PromotionCode) VALUES (2, 101, 201, 301)");
        databaseConnection.executeQuery("INSERT INTO INCLUDES (ReservationID,
PCCode, ASCode, PromotionCode) VALUES (3, 101, 202, 649052)");
        databaseConnection.executeQuery("INSERT INTO INCLUDES (ReservationID,
PCCode, ASCode, PromotionCode) VALUES (4, 102, 203, 649052)");
        databaseConnection.executeQuery("INSERT INTO INCLUDES (ReservationID,
PCCode, ASCode, PromotionCode) VALUES (5, 103, 204, 301)");

        databaseConnection.executeQuery("INSERT INTO CREATES (ReservationID,
BillingID) VALUES (1, 1)");
        databaseConnection.executeQuery("INSERT INTO CREATES (ReservationID,
BillingID) VALUES (2, 2)");
        databaseConnection.executeQuery("INSERT INTO CREATES (ReservationID,
BillingID) VALUES (3, 3)");
            databaseConnection.executeQuery("INSERT INTO CREATES (ReservationID,
BillingID) VALUES (4, 4)");
            databaseConnection.executeQuery("INSERT INTO CREATES (ReservationID,
BillingID) VALUES (5, 5)");

            JOptionPane.showMessageDialog(mainMenuFrame, "All Tables Populated!");
        });
    }
}
```

ViewTables.java

```
import java.awt.EventQueue;
import java.awt.Font;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class ViewTables {

    private JFrame viewTableFrame;
    DatabaseConnection databaseConnection = new DatabaseConnection();

    private static List<String> queryNames = new ArrayList<>();
    private static List<String> queryCommands = new ArrayList<>();

    public static void viewTableRunnable() {
        EventQueue.invokeLater(() -> {
            try {
                ViewTables viewTableWindow = new ViewTables();
                viewTableWindow.viewTableFrame.setVisible(true);
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    public ViewTables() {
        if (databaseConnection.createConnection())  {
            createViewTableWindow();
        }
    }
```

```
private void createViewTableWindow() {
    viewTableFrame = new JFrame();
    viewTableFrame.setTitle("View Tables");
    viewTableFrame.setBounds(100, 100, 360, 500);
    viewTableFrame.getContentPane().setLayout(null);
    viewTableFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel carRentalSystemTableLabel = new JLabel("Tables");
    carRentalSystemTableLabel.setFont(new Font("Garamond", Font.BOLD, 24));
    carRentalSystemTableLabel.setBounds(10, 0, 300, 50);
    viewTableFrame.getContentPane().add(carRentalSystemTableLabel);

    queryNames.add("Reservation");
    queryNames.add("Billing");
    queryNames.add("Location");
    queryNames.add("Car");
    queryNames.add("Car Model");
    queryNames.add("Car Type");
    queryNames.add("Protections & Coverages");
    queryNames.add("Accessories & Services");
    queryNames.add("Promotions");
    queryNames.add("Customer");

    queryCommands.add("SELECT * FROM RESERVATION");
    queryCommands.add("SELECT * FROM BILLING");
    queryCommands.add("SELECT * FROM LOCATION");
    queryCommands.add("SELECT * FROM CAR");
    queryCommands.add("SELECT * FROM CAR_MODEL");
    queryCommands.add("SELECT * FROM CAR_TYPE");
    queryCommands.add("SELECT * FROM PROTECTIONS_COVERAGES");
    queryCommands.add("SELECT * FROM ACCESSORIES_SERVICES");
    queryCommands.add("SELECT * FROM PROMOTION");
    queryCommands.add("SELECT * FROM CUSTOMER");

    JButton reservationTableButton = new JButton(queryNames.get(0));
    reservationTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
    reservationTableButton.setBounds(50, 50, 240, 30);
    viewTableFrame.getContentPane().add(reservationTableButton);

    JButton billingTableButton = new JButton(queryNames.get(1));
    billingTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
    billingTableButton.setBounds(50, 90, 240, 30);
    viewTableFrame.getContentPane().add(billingTableButton);

    JButton locationTableButton = new JButton(queryNames.get(2));
    locationTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
    locationTableButton.setBounds(50, 170, 240, 30);
    viewTableFrame.getContentPane().add(locationTableButton);
```

```
        JButton carTableButton = new JButton(queryNames.get(3));
        carTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        carTableButton.setBounds(50, 130, 240, 30);
        viewTableFrame.getContentPane().add(carTableButton);

        JButton carModelTableButton = new JButton(queryNames.get(4));
        carModelTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        carModelTableButton.setBounds(50, 210, 240, 30);
        viewTableFrame.getContentPane().add(carModelTableButton);

        JButton carTypeTableButton = new JButton(queryNames.get(5));
        carTypeTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        carTypeTableButton.setBounds(50, 250, 240, 30);
        viewTableFrame.getContentPane().add(carTypeTableButton);

        JButton protectionsCoveragesTableButton = new JButton(queryNames.get(6));
        protectionsCoveragesTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        protectionsCoveragesTableButton.setBounds(50, 290, 240, 30);
        viewTableFrame.getContentPane().add(protectionsCoveragesTableButton);

        JButton accessoriesServicesTableButton = new JButton(queryNames.get(7));
        accessoriesServicesTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        accessoriesServicesTableButton.setBounds(50, 330, 240, 30);
        viewTableFrame.getContentPane().add(accessoriesServicesTableButton);

        JButton promotionsTableButton = new JButton(queryNames.get(8));
        promotionsTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        promotionsTableButton.setBounds(50, 370, 240, 30);
        viewTableFrame.getContentPane().add(promotionsTableButton);

        JButton customerTableButton = new JButton(queryNames.get(9));
        customerTableButton.setFont(new Font("Garamond", Font.BOLD, 18));
        customerTableButton.setBounds(50, 410, 240, 30);
        viewTableFrame.getContentPane().add(customerTableButton);

        Menu.executeButtonActionEvent(reservationTableButton, databaseConnection,
queryCommands.get(0));
        Menu.executeButtonActionEvent(billingTableButton, databaseConnection,
queryCommands.get(1));
        Menu.executeButtonActionEvent(locationTableButton, databaseConnection,
queryCommands.get(2));
        Menu.executeButtonActionEvent(carTableButton, databaseConnection,
queryCommands.get(3));
        Menu.executeButtonActionEvent(carModelTableButton, databaseConnection,
queryCommands.get(4));
        Menu.executeButtonActionEvent(carTypeTableButton, databaseConnection,
queryCommands.get(5));
```

```
        Menu.executeButtonActionEvent(protectionsCoveragesTableButton,
databaseConnection, queryCommands.get(6));
        Menu.executeButtonActionEvent(accessoriesServicesTableButton, databaseConnection,
queryCommands.get(7));
        Menu.executeButtonActionEvent(promotionsTableButton, databaseConnection,
queryCommands.get(8));
        Menu.executeButtonActionEvent(customerTableButton, databaseConnection,
queryCommands.get(9));

        JButton backButton = new JButton("←");
        backButton.setFont(new Font("Garamond", Font.BOLD, 14));
        backButton.setBounds(280, 15, 50, 20);
        viewTableFrame.getContentPane().add(backButton);

        backButton.addActionListener(actionEvent -> {
            Menu menu = new Menu();
            menu.menuRunnable();
            viewTableFrame.dispose();
        });
    }
}
```

QueryTables.java

```
import java.awt.EventQueue;
import java.awt.Font;
import java.util.ArrayList;
import java.util.List;
import javax.swing.*;

public class QueryTables {

    private JFrame queryTablesFrame;
    private DatabaseConnection databaseConnection = new DatabaseConnection();

    private static List<String> queryNames = new ArrayList<>();
    private static List<String> queryCommands = new ArrayList<>();

    public static void queryTableRunnable() {
        EventQueue.invokeLater(() -> {
            try {
                QueryTables queryTablesWindow = new QueryTables();
                queryTablesWindow.queryTablesFrame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }
```

```
  public QueryTables() {
     if (databaseConnection.createConnection()) {
        createQueryTablesWindow();
     }
  }

  private void createQueryTablesWindow() {
     queryTablesFrame = new JFrame();
     queryTablesFrame.setTitle("Query Tables");
     queryTablesFrame.setBounds(100, 100, 480, 550);
     queryTablesFrame.getContentPane().setLayout(null);
     queryTablesFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

     JLabel carRentalSystemQueryLabel = new JLabel("Common Queries");
     carRentalSystemQueryLabel.setFont(new Font("Garamond", Font.BOLD, 24));
     carRentalSystemQueryLabel.setBounds(10, 0, 300, 50);
     queryTablesFrame.getContentPane().add(carRentalSystemQueryLabel);

     queryNames.add("1. List of Cars With More Than 7 Seats");
     queryNames.add("2. List of Promotions That Are Less Than Or Equal To 70");
     queryNames.add("3. List of Locations With LocationNumber Over 5000");
     queryNames.add("4. List of Promotions That Are Percentages");
     queryNames.add("5. List of Promotions That Are Amounts");
     queryNames.add("6. List of Cars Whose Mileage Is Over 10000");
     queryNames.add("7. List of Members From Toronto");
     queryNames.add("8. List of Cars That Are Honda Or Toyota");

     queryCommands.add("SELECT DISTINCT cm.ModelName, cm.Manufacturer,
cm.ModelYear, cm.NumberOfSeats FROM CAR c, CAR_MODEL cm WHERE
cm.NumberOfSeats >= 7 and c.ModelNumber = cm.ModelNumber");
     queryCommands.add("SELECT * FROM PROMOTION_AMOUNT WHERE Amount <=
70");
     queryCommands.add("SELECT DISTINCT LocationName, l.LocationNumber,
PhoneNumber FROM LOCATION l, HAS h WHERE l.LocationNumber = h.LocationNumber
AND h.LocationNumber >= 5000");
     queryCommands.add("SELECT DISTINCT PROMOTION.PromotionName,
PROMOTION.PromotionCode, PROMOTION_PERCENTAGE.Percentage FROM
PROMOTION_PERCENTAGE, PROMOTION WHERE PROMOTION.PromotionCode =
PROMOTION_PERCENTAGE.PromotionCode");
     queryCommands.add("SELECT DISTINCT PROMOTION.PromotionName,
PROMOTION.PromotionCode, PROMOTION_AMOUNT.Amount FROM
PROMOTION_AMOUNT, PROMOTION WHERE PROMOTION.PromotionCode =
PROMOTION_AMOUNT.PromotionCode");
     queryCommands.add("SELECT * FROM CAR WHERE Mileage >= 100000");
     queryCommands.add("SELECT DISTINCT cu.FirstName, cu.LastName,
cu.EmailAddress, cu.PhoneNumber FROM MEMBER m, CUSTOMER cu WHERE City =
'Toronto' AND m.LicenseNumber = cu.LicenseNumber");
```

```
        queryCommands.add("SELECT DISTINCT cm.ModelName, cm.Manufacturer,
cm.ModelYear FROM CAR c, CAR_MODEL cm WHERE cm.Manufacturer = 'Toyota' OR
cm.Manufacturer = 'Honda' AND c.ModelNumber = cm.ModelNumber");

        JButton queryOneButton = new JButton(queryNames.get(0));
        queryOneButton.setFont(new Font("Garamond", Font.BOLD, 14));
        queryOneButton.setBounds(30, 60, 400, 30);
        queryTablesFrame.getContentPane().add(queryOneButton);

        JButton queryTwoButton = new JButton(queryNames.get(1));
        queryTwoButton.setFont(new Font("Garamond", Font.BOLD, 14));
        queryTwoButton.setBounds(30, 100, 400, 30);
        queryTablesFrame.getContentPane().add(queryTwoButton);

        JButton queryThreeButton = new JButton(queryNames.get(2));
        queryThreeButton.setFont(new Font("Garamond", Font.BOLD, 14));
        queryThreeButton.setBounds(30, 140, 400, 30);
        queryTablesFrame.getContentPane().add(queryThreeButton);

        JButton queryFourButton = new JButton(queryNames.get(3));
        queryFourButton.setFont(new Font("Garamond", Font.BOLD, 14));
        queryFourButton.setBounds(30, 180, 400, 30);
        queryTablesFrame.getContentPane().add(queryFourButton);

        JButton queryFiveButton = new JButton(queryNames.get(4));
        queryFiveButton.setFont(new Font("Garamond", Font.BOLD, 14));
        queryFiveButton.setBounds(30, 220, 400, 30);
        queryTablesFrame.getContentPane().add(queryFiveButton);

        JButton querySixButton = new JButton(queryNames.get(5));
        querySixButton.setFont(new Font("Garamond", Font.BOLD, 14));
        querySixButton.setBounds(30, 260, 400, 30);
        queryTablesFrame.getContentPane().add(querySixButton);

        JButton querySevenButton = new JButton(queryNames.get(6));
        querySevenButton.setFont(new Font("Garamond", Font.BOLD, 14));
        querySevenButton.setBounds(30, 300, 400, 30);
        queryTablesFrame.getContentPane().add(querySevenButton);

        JButton queryEightButton = new JButton(queryNames.get(7));
        queryEightButton.setFont(new Font("Garamond", Font.BOLD, 14));
        queryEightButton.setBounds(30, 340, 400, 30);
        queryTablesFrame.getContentPane().add(queryEightButton);

        Menu.executeButtonActionEvent(queryOneButton, databaseConnection,
queryCommands.get(0));
        Menu.executeButtonActionEvent(queryTwoButton, databaseConnection,
queryCommands.get(1));
```

```
      Menu.executeButtonActionEvent(queryThreeButton, databaseConnection,
queryCommands.get(2));
      Menu.executeButtonActionEvent(queryFourButton, databaseConnection,
queryCommands.get(3));
      Menu.executeButtonActionEvent(queryFiveButton, databaseConnection,
queryCommands.get(4));
      Menu.executeButtonActionEvent(querySixButton, databaseConnection,
queryCommands.get(5));
      Menu.executeButtonActionEvent(querySevenButton, databaseConnection,
queryCommands.get(6));
      Menu.executeButtonActionEvent(queryEightButton, databaseConnection,
queryCommands.get(7));

      JButton customQueryButton = new JButton("Custom Query");
      customQueryButton.setFont(new Font("Garamond", Font.BOLD, 12));
      customQueryButton.setBounds(30, 380, 120, 30);
      queryTablesFrame.getContentPane().add(customQueryButton);

      JTextField customQueryTextField = new JTextField();
      customQueryTextField.setBounds(160, 380, 260, 100);
      customQueryTextField.setColumns(1000);
      queryTablesFrame.getContentPane().add(customQueryTextField);

      customQueryButton.addActionListener(actionEvent -> {
         String customQuery = customQueryTextField.getText();
         Menu.executeButtonActionEvent(customQueryButton, databaseConnection,
customQuery);
      });

      JButton backButton = new JButton("←");
      backButton.setFont(new Font("Garamond", Font.BOLD, 12));
      backButton.setBounds(400, 15, 50, 20);
      queryTablesFrame.getContentPane().add(backButton);

      backButton.addActionListener(actionEvent -> {
         Menu menu = new Menu();
         menu.menuRunnable();
         queryTablesFrame.dispose();
      });
   }
}
```

UpdateTables.java

```
import java.awt.EventQueue;
import java.awt.Font;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class UpdateTables {

  private JFrame updateTableFrame;
  DatabaseConnection databaseConnection = new DatabaseConnection();

  public void updateTablesRunnable() {
     EventQueue.invokeLater(() -> {
        try {
           UpdateTables updateTableWindow = new UpdateTables();
           updateTableWindow.updateTableFrame.setVisible(true);
        }
        catch (Exception e) {
           e.printStackTrace();
        }
     });
  }

  public UpdateTables() {
     if (databaseConnection.createConnection()) {
        createUpdateTablesWindow();
     }
  }

  private void createUpdateTablesWindow() {
     updateTableFrame = new JFrame();
     updateTableFrame.setBounds(100, 100, 480, 320);
     updateTableFrame.getContentPane().setLayout(null);
     updateTableFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

     JLabel carRentalEditLabel = new JLabel("Edit Table");
     carRentalEditLabel.setFont(new Font("Garamond", Font.BOLD, 24));
     carRentalEditLabel.setBounds(10, 0, 300, 50);
     updateTableFrame.getContentPane().add(carRentalEditLabel);

     JButton updateButton = new JButton("Update");
     updateButton.setFont(new Font("Garamond", Font.BOLD, 16));
     updateButton.setBounds(20, 60, 100, 30);
     updateTableFrame.getContentPane().add(updateButton);
```

```
    JTextField updateTextField = new JTextField("UPDATE table-name");
    updateTextField.setBounds(140 , 60, 300, 20);
    updateTextField.setColumns(500);
    updateTableFrame.getContentPane().add(updateTextField);

    JTextField setTextField = new JTextField("SET column-name = value, column-name =
value, ...");
    setTextField.setBounds(140 , 80, 300, 20);
    setTextField.setColumns(500);
    updateTableFrame.getContentPane().add(setTextField);

    JTextField whereTextField = new JTextField("WHERE condition");
    whereTextField.setBounds(140 , 100, 300, 20);
    whereTextField.setColumns(500);
    updateTableFrame.getContentPane().add(whereTextField);

    updateButton.addActionListener(actionEvent -> {
        String updateTextString = updateTextField.getText();
        String setTextString = setTextField.getText();
        String whereTextString = whereTextField.getText();

        String queryStatement = updateTextString + " " + setTextString + " " +
whereTextString;
        databaseConnection.executeQuery(queryStatement);

        JOptionPane.showMessageDialog(updateTableFrame, "Updated!");
    });

    JButton insertButton = new JButton("Insert");
    insertButton.setFont(new Font("Garamond", Font.BOLD, 16));
    insertButton.setBounds(20, 140, 100, 30);
    updateTableFrame.getContentPane().add(insertButton);

    JTextField insertIntoTextField = new JTextField("INSERT INTO table-name");
    insertIntoTextField.setBounds(140 , 140, 300, 20);
    insertIntoTextField.setColumns(500);
    updateTableFrame.getContentPane().add(insertIntoTextField);

    JTextField insertAttributeNamesTextField = new JTextField("(column-names)");
    insertAttributeNamesTextField.setBounds(140 , 160, 300, 20);
    insertAttributeNamesTextField.setColumns(500);
    updateTableFrame.getContentPane().add(insertAttributeNamesTextField);

    JTextField insertValuesTextField = new JTextField("VALUES (values)");
    insertValuesTextField.setBounds(140 , 180, 300, 20);
    insertValuesTextField.setColumns(500);
    updateTableFrame.getContentPane().add(insertValuesTextField);
    insertButton.addActionListener(actionEvent -> {
      String insertIntoString = insertIntoTextField.getText();
      String insertAttributeNamesString = insertAttributeNamesTextField.getText();
      String insertValuesString = insertValuesTextField.getText();

      String queryStatement = insertIntoString + " " + insertAttributeNamesString + " " +
insertValuesString;
      databaseConnection.executeQuery(queryStatement);

      JOptionPane.showMessageDialog(insertButton, "Inserted!");
    });

    JButton removeButton = new JButton("Remove");
    removeButton.setFont(new Font("Garamond", Font.BOLD, 16));
    removeButton.setBounds(20, 220, 100, 30);
    updateTableFrame.getContentPane().add(removeButton);

    JTextField deleteTableTextField = new JTextField("DELETE table-name");
    deleteTableTextField.setBounds(140, 220, 300, 20);
    deleteTableTextField.setColumns(500);
    updateTableFrame.getContentPane().add(deleteTableTextField);

    JTextField deleteConditionTextField = new JTextField("WHERE condition");
    deleteConditionTextField.setBounds(140, 240, 300, 20);
    deleteConditionTextField.setColumns(500);
    updateTableFrame.getContentPane().add(deleteConditionTextField);

    removeButton.addActionListener(actionEvent -> {
      String deleteTableString = deleteTableTextField.getText();
      String deleteConditionString = deleteConditionTextField.getText();

      String queryStatement = deleteTableString + " " + deleteConditionString;
      databaseConnection.executeQuery(queryStatement);

      JOptionPane.showMessageDialog(removeButton, "Removed!");
    });

    JButton backButton = new JButton("←");
    backButton.setFont(new Font("Garamond", Font.BOLD, 12));
    backButton.setBounds(400, 15, 50, 20);
    updateTableFrame.getContentPane().add(backButton);

    backButton.addActionListener(actionEvent -> {
      Menu menu = new Menu();
      menu.menuRunnable();
      updateTableFrame.dispose();
    });
  }}
```

Relational Algebra For Queries

1. List of customers that live in London.

```
SELECT *
FROM CUSTOMER
WHERE City = 'London'
```

$$\sigma_{city='London'}(Customer)$$

2. List of members whose membership ID is 1002.

```
SELECT *
FROM MEMBER
WHERE MembershipID = 1002
```

$$\sigma_{MembershipID=1002}(Member)$$

3. List the ReservationID, StartDate, LocationName, StreetAddress, PhoneNumber information for all reservations.

```
SELECT ReservationID, StartDate, LocationName, StreetAddress, PhoneNumber
FROM RESERVATION r, LOCATION l
WHERE r.PickupLocation = l.LocationNumber
```

$$\pi_{ReservationID,\ StartDate,\ LocationName,\ StreetAddress,\ PhoneNumber}(\sigma_{Reservation.PickupLocation\ =\ Location.LocationNumber}(Reservation \bowtie Location))$$

4. List of ModelName and Mileage for cars whose manfacturer is Honda.

```
SELECT cm.modelname, car.mileage
FROM CAR car, car_model cm, RESERVATION reservation, chosen
WHERE
        cm.manufacturer = 'Honda'
AND    car.modelnumber = cm.modelnumber
AND    car.registrationnumber = chosen.registrationnumber
AND    chosen.reservationid = reservation.reservationid
```

$$\pi_{ModelName,\ Mileage}(\sigma_{Manufacturer=Honda'}(Car \bowtie Car\_Model \bowtie Reservation \bowtie Chosen))$$

5. List the ModelName and Milege for cars of type 'Truck' whose mileage is not over 100000.

SELECT cm.modelname, Mileage
FROM car_model cm, CAR ca, car_type ct
WHERE
        ct.cartypename = 'Truck'
AND   cm.modelnumber = ca.modelnumber AND NOT ca.mileage < 100000

$\pi_{\text{ModelName, Mileage}}$ $(\sigma_{\text{CarTypeName='Truck' AND NOT (Mileage<100000)}}(\text{Car\_Model} \bowtie \text{Car} \bowtie \text{Car\_Type}))$

6. List all the car types.

SELECT *
FROM CAR_TYPE
	$\sigma(\text{Car\_Type})$

7. List the LocationName and PhoneNumber that have the LocationNumber equal to or bigger than 5000.

SELECT distinct LocationName, PhoneNumber
FROM LOCATION l , HAS h
WHERE l.LocationNumber = h.LocationNumber
AND   h.LocationNumber >= 5000

$\pi_{\text{locationNumber, PhoneNumber}}(\sigma_{\text{LocationNumber >= 5000}} (\text{Location} \bowtie \text{Has}))$

8. List the max total fees and average total fees.

SELECT MAX(carrentalfees) as MaxTotalFee,
AVG(carrentalfees) as AverageTotalFee
FROM BILLING;
	$\rho_{\text{MaxTotalFee, AverageTotalFee}}(F_{\text{MAX CarRentalFees, AVG CarRentalFees}}(\text{Billing}))$

9. List of the accessories and services (in descending order) where the price is equal to 10.

SELECT *
FROM ACCESSORIES_SERVICES
WHERE ASPrice = 10
ORDER BY AsCode DESC
	τ **AsCode desc** σ **ASPrice = 10**
	(Accessories_Services)

10. List the promotions that have the name ‘First Rental’.

SELECT *
FROM PROMOTION
WHERE PromotionName = 'First Rental'
	$\sigma_{\text{promotionName = ‘First Rental’}}$ **(Promotion)**

11. List the promotion amounts that are equal to or less than 70.

SELECT *
FROM PROMOTION_AMOUNT
WHERE Amount <= 70
	$\sigma_{\text{Amount <= 70}}$ **(Promotion_Amount)**

12. List the billing information for itms where the billing status is ‘BAD’ and DiscountValue is not NULL or that carRentalFees is equal to or bigger than 300 and AsFees is equal to or bigger than 50.

SELECT *
from BILLING
WHERE (BillingStatus = 'BAD'
AND NOT(DiscountValue = Null))
OR (carrentalfees >=300 and AsFees >=50)

$\text{Temp1} \leftarrow \sigma_{\text{BillingStatus = ‘BAD’ AND NOT (DiscountValue = NULL)}}(\text{Billing})$ $\text{Temp2} \leftarrow \sigma_{\text{CarRentalFees >= 300 AND AsFees >= 50}}(\text{Billing})$ $\text{Result} \leftarrow \text{Temp1} \cup \text{Temp2}$

CONCLUSION

The creation of this Car Rental Database Management System, from concept to creation, has helped us to understand the different aspects of the database system. By incrementally developing our database through the weeks, it helped us understand the process of creating a fully-functoning database. The skills that we learned through the weeks, as we developed this database, are applicable to the real-world and we are confident that we would be able to work with databases in the future.