CPS510 Project

Car Rental DBMS

Kathleen Furtado (500761526)
Austin Cheung (500810580)
Paul Guevarra (500833529)

Table of Contents

1 - Application Description

Car Rental System "*EZ–PZ Rentals*"
A database system for car rental dealerships to use in their day-to-day operations to create and manage rental bookings. *EZ–PZ Rentals* has a number of different locations offering a variety of vehicles. Employees using the system can create client accounts and bookings with a vehicle for a specified number of days. Each vehicle has a unique ID, is stored at one of our 3 locations, and has daily rental costs associated with it. A client, transaction, and vehicle will be associated with each booking.

Vehicle – This is an entity is for all the vehicles that can be rented.
- Vehicle ID
- Number of seats
- Color
- Daily Rate
- Model
- Car Location

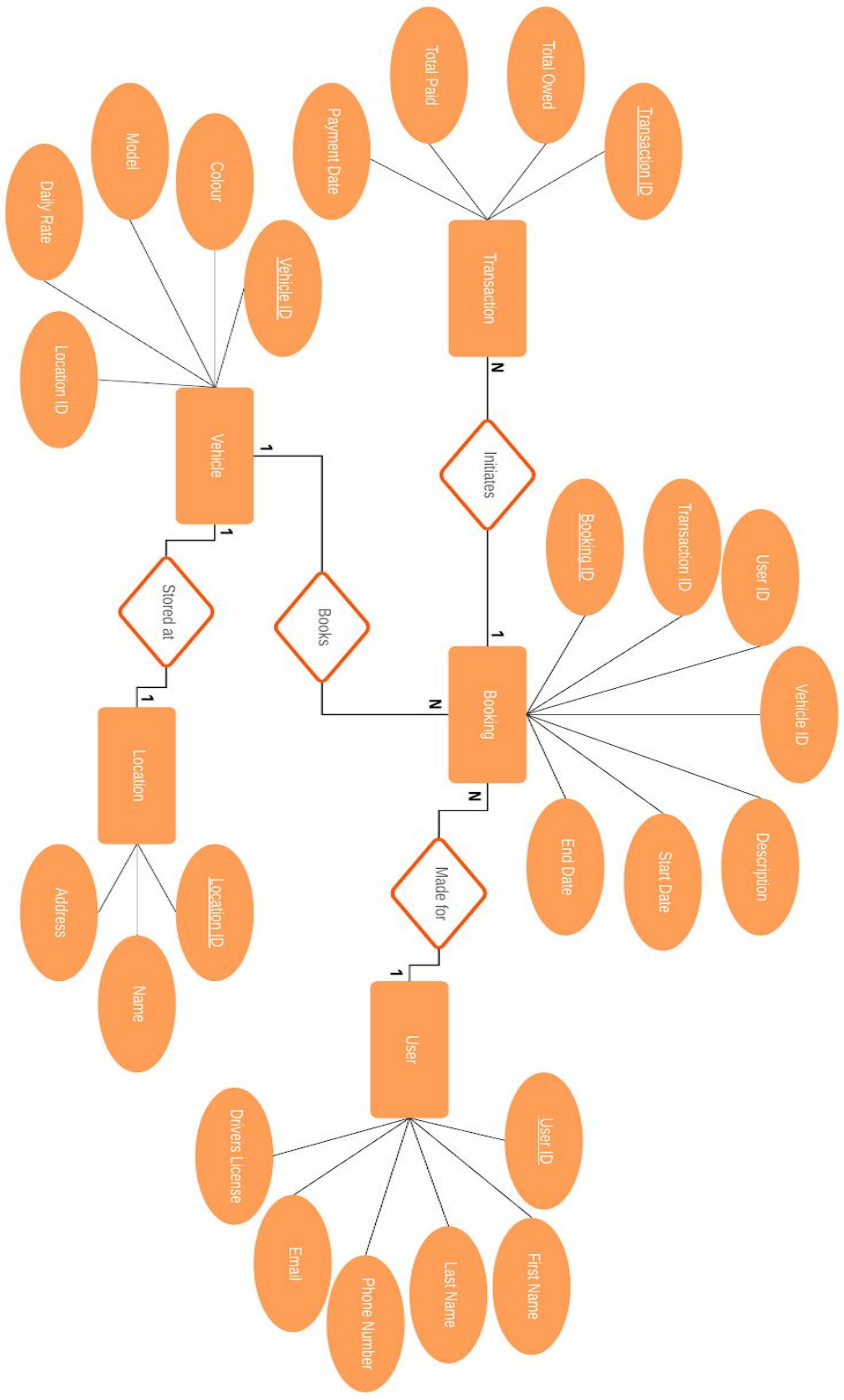
Transaction – This includes information regarding billing for each rental.
- Transaction ID
- Total owed
- Total payed
- Payment date
-

User Account – This is the user profile with the company.
- User ID
- Email
- Phone number
- Driver's license number
- First Name
- Last Name

Booking – Each booking includes the related client, car, and transaction, as well as the relevant dates
- Booking ID
- Transaction ID
- Vehicle ID
- User ID
- Description
- Start date
- End date

Location – This describes the different locations of the company and where each vehicle is stored.
- Location ID
- Name
- Address

2 - ER Diagram


Transaction
Payment Date
Total Paid
Total Owed
Transaction ID
N
Initiates
1
Booking
Booking ID
Transaction ID
User ID
Vehicle ID
Description
Start Date
End Date
N
Books
1
Vehicle
Daily Rate
Model
Colour
Vehicle ID
Location ID
1
Stored at
1
Location
Address
Name
Location ID
N
Made for
1
User
Drivers License
Email
Phone Number
Last Name
First Name
User ID

3 - Schema Design

```
CREATE TABLE locations (
LocationID     NUMBER PRIMARY KEY NOT NULL,
Name           VARCHAR2(20) NOT NULL,
Address        VARCHAR2(20) NOT NULL);

CREATE TABLE userAccount (
UserID         Number PRIMARY KEY NOT NULL,
Email          VARCHAR2(255) NOT NULL,
PhoneNumber    VARCHAR2(10),
DriverLicense VARCHAR2(15) NOT NULL UNIQUE,
FirstName      VARCHAR2(255) NOT NULL,
LastName       VARCHAR2(255) NOT NULL);

CREATE TABLE transactions (
TransactionID NUMBER PRIMARY KEY NOT NULL,
TotalOwed      NUMBER NOT NULL,
TotalPaid      NUMBER DEFAULT 0 NOT NULL,
PaymentDate    DATE);

CREATE TABLE vehicle (
VehicleID      NUMBER PRIMARY KEY NOT NULL,
NumOfSeats     NUMBER NOT NULL,
Colour         VARCHAR2(20) NOT NULL,
DailyRate      NUMBER NOT NULL,
CarModel       VARCHAR2(20) NOT NULL,
CarLocation    NUMBER REFERENCES locations(LocationID));

CREATE TABLE booking (
BookingID      NUMBER PRIMARY KEY NOT NULL,
TransactionID NUMBER REFERENCES transactions(transactionID),
VehicleID      NUMBER REFERENCES Vehicle(VehicleID),
UserID         NUMBER REFERENCES userAccount(userID),
Description    VARCHAR2(255),
StartDate      DATE NOT NULL,
EndDate        DATE NOT NULL);
```

4 - Simple Queries

```
SELECT vehicleID
FROM vehicle
WHERE seats > 5;
```
$\sigma_{\text{numofSeats} > 5}(\text{vehicle})$

```
SELECT vehicleID, colour, carmodel, numofseats
FROM vehicle
WHERE carlocation = 1;
```
$\pi_{\text{vehicleID, colour, carmodel,numofseats}}\ (\sigma_{\text{carlocation}= 1}(\text{vehicle}))$

```
SELECT email, phonenumber
FROM USERACCOUNT
WHERE firstname = 'Alex';
```
$\pi_{\text{email, phonenumber}}\ (\sigma_{\text{firstname = Alex}}\ (\text{userAccount}))$

```
SELECT *
FROM transactions
WHERE totalowed > 0;
```
$\sigma_{\text{totalOwed>0}}(\text{transactions})$

```
SELECT booking.bookingID
FROM booking, useraccount
WHERE useraccount.firstname = 'Alex'
AND booking.userid = useraccount.userID;
```
$\pi_{\text{bookingID}}(\sigma_{\text{firstname = Alex}}\ (\text{userAccount})\hat{}\ \sigma_{\text{userID}}\ (\text{booking})_{=\ \text{userID}}(\text{userAccount})\ )$

```
SELECT bookingID
FROM booking
WHERE vehicleID = 12345
```
$\sigma_{\text{bookingID}}(_{\text{vehicleID = 12345}}(\text{booking}))$

```
SELECT firstname, lastname, email, phonenumber
FROM useraccount;
```
$\pi_{\text{firstname,lastname,email, phonenumber}}\ (\text{userAccount})$

5 - Advanced Queries (Unix shell implementation)

Creating tables:

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"ksfurtad/********@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(
Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

CREATE TABLE locations (
LocationID     NUMBER PRIMARY KEY NOT NULL,
Name           VARCHAR2(20) NOT NULL,
Address        VARCHAR2(20) NOT NULL);

CREATE TABLE userAccount (
userID        Number PRIMARY KEY NOT NULL,
Email         VARCHAR2(255) NOT NULL,
PhoneNumber   VARCHAR2(10),
DriverLicense VARCHAR2(15) NOT NULL UNIQUE,
FirstName     VARCHAR2(255) NOT NULL,
LastName      VARCHAR2(255) NOT NULL);

CREATE TABLE transactions (
TransactionID NUMBER PRIMARY KEY NOT NULL,
TotalOwed     NUMBER NOT NULL,
TotalPaid     NUMBER DEFAULT 0 NOT NULL,
PaymentDate   DATE);

CREATE TABLE vehicle (
VehicleID     NUMBER PRIMARY KEY NOT NULL,
NumOfSeats    NUMBER NOT NULL,
Colour        VARCHAR2(20) NOT NULL,
DailyRate     NUMBER NOT NULL,
CarModel      VARCHAR2(20) NOT NULL,
CarLocation   NUMBER REFERENCES locations(LocationID));

CREATE TABLE booking (
BookingID     NUMBER PRIMARY KEY NOT NULL,
TransactionID NUMBER REFERENCES transactions(transactionID),
VehicleID     NUMBER REFERENCES Vehicle(VehicleID),
UserID        NUMBER REFERENCES userAccount(userID),
Description   VARCHAR2(255),
StartDate     DATE NOT NULL,
EndDate       DATE NOT NULL);

exit;
EOF
```

Advanced Queries:

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"ksfurtad/********@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(
Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

-- Cars available today:
SELECT vehicle.vehicleID, vehicle.colour, vehicle.numofseats
FROM booking JOIN vehicle
ON booking.vehicleID = vehicle.vehicleID
WHERE not(sysdate between booking.startdate and booking.enddate);
```

-- $\pi_{vehicleID,color,numofseats}(booking \bowtie vehicle)(\sigma_{vehicleID}(booking)_{=}$

--$_{vehicleID}(userAccount)\wedge(_{sysDate!>startDate}\wedge_{sysDate!<endDate})$

```
-- All bookings
SELECT *
FROM booking JOIN vehicle
ON booking.vehicleID = vehicle.vehicleID;
```

-- $\sigma(_{vehicleID}(booking)_{=\ vehicleID}(userAccount))\ (booking \bowtie vehicle)$

```
-- Bookings that start this week
SELECT *
FROM booking JOIN vehicle
ON booking.vehicleID = vehicle.vehicleID
WHERE EXTRACT(DAY FROM booking.startdate) >=20;
```

-- $\sigma(_{vehicleID}(booking)_{=\ vehicleID}(userAccount))\ (booking \bowtie vehicle)\ (\sigma_{Day}(booking) >=20)$

```
-- Outstanding balances:
Select useraccount.firstname, useraccount.lastname, transactions.totalowed,
transactions.TransactionID
FROM transactions, booking, USERACCOUNT
WHERE booking.transactionID = transactions.TransactionID
AND booking.userID = useraccount.userID
AND transactions.totalowed >0;
```

-- $\pi_{firstName,lastName,totalowed,transactionID}(transactions \bowtie booking \bowtie useraccount)(\sigma_{transactionID}(booking)_{=\ transactionID}$

--$(booking)\wedge\sigma_{userID}(booking)_{=\ userID}(useraccount)\wedge_{totalowed}(transactions)_{>0})$

```
-- Full paid customers:
Select useraccount.firstname, useraccount.lastname, transactions.totalowed,
transactions.TransactionID
FROM transactions, booking, USERACCOUNT
WHERE booking.transactionID = transactions.TransactionID
AND booking.userID = useraccount.userID
AND transactions.totalowed =0;
```

-- $\pi_{firstName,lastName,totalowed,transactionID}(transactions \bowtie booking \bowtie useraccount)(\sigma_{transactionID}(booking)_{=\ transactionID}$

--$(booking\wedge\sigma_{userID}(booking)_{=\ userID}(useraccount)\wedge_{totalowed}(transactions) = 0)$

```
exit;
EOF
```

Drop Tables:

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"ksfurtad/********@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(
Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
DROP TABLE VEHICLE CASCADE CONSTRAINTS;
DROP TABLE USERACCOUNT CASCADE CONSTRAINTS;
DROP TABLE LOCATIONS CASCADE CONSTRAINTS;
DROP TABLE BOOKING CASCADE CONSTRAINTS;
DROP TABLE TRANSACTIONS CASCADE CONSTRAINTS;
exit;
EOF
```

Insert Data:

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"ksfurtad/*******@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(P
ort=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
INSERT into locations (LocationID, Name, Address)
Values(1,'toronto', '14 Database Rd');
INSERT into locations (LocationID, Name, Address)
Values(2,'scarborough', '341 Oracle Lane');
INSERT into locations (LocationID, Name, Address)
Values(3,'etobicoke', '8644 Data Crt');

INSERT into vehicle (VehicleID, NumOfSeats, Colour, DailyRate, CarModel,
CarLocation)
VALUES(12345,7,'red',74.56,'Nissan',1);
INSERT into vehicle (VehicleID, NumOfSeats, Colour, DailyRate, CarModel,
CarLocation)
VALUES(12346,5,'black',86.00,'Honda',2);
INSERT into vehicle (VehicleID, NumOfSeats, Colour, DailyRate, CarModel,
CarLocation)
VALUES(12347,5,'black',95.45,'Toyota',3);
INSERT into vehicle (VehicleID, NumOfSeats, Colour, DailyRate, CarModel,
CarLocation)
VALUES(12348,7,'silver',250.00,'Ford',3);

INSERT into userAccount (userID, email, phonenumber,DriverLicense, FirstName,
LastName)
VALUES(12345,'jordanlai@gmail.com',4165469987,2947592273, 'Jordan','Lai');
INSERT into userAccount (userID, email, phonenumber,DriverLicense, FirstName,
LastName)
VALUES(12346,'rebecca_smith@hotmail.com',6478935529,968833610, 'Rebecca','Smith');
INSERT into userAccount (userID, email, phonenumber,DriverLicense, FirstName,
LastName)
VALUES(12347,'alexaldea3@hotmail.com',6471129472,1237384552, 'Alex','Aldea');

INSERT into transactions (TransactionID,TotalOwed,TotalPaid,PaymentDate)
VALUES(67, 23.95, 233.78, '2019-10-24');
INSERT into booking (BookingID, TransactionID, VehicleID, userID, Description,
StartDate, EndDate)
```

```
VALUES(34,67,12348,12345,'deposit has been paid.','2019-10-25', '2019-10-27');

INSERT into transactions (TransactionID,TotalOwed,TotalPaid,PaymentDate)
VALUES(56,88.56, 34.23, '2019-10-22');
INSERT into booking (BookingID, TransactionID, VehicleID,userID, Description,
StartDate, EndDate)
VALUES(45,56,12346,12346,'deposit has been paid.','2019-10-26', '2019-10-29');

INSERT into transactions (TransactionID,TotalOwed,TotalPaid,PaymentDate)
VALUES(81, 0, 76.55, '2019-10-22');
INSERT into booking (BookingID, TransactionID, VehicleID,userID, Description,
StartDate, EndDate)
VALUES(42, 81,12345, 12347,'Paid in full.','2019-10-29', '2019-10-31');

exit;
EOF
```

6 - Functional Dependencies

Table: **locations**
LocationID -> name, address

Table: **userAccount**
Username -> password, email, phoneNumber, driversLicense, firstName, lastName

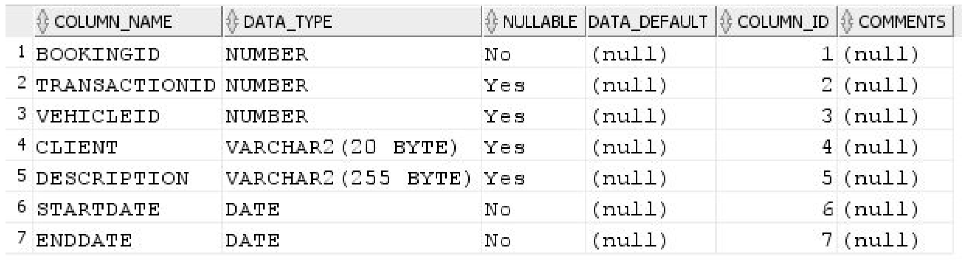
Table: **transactions**
TransactionID -> totalOwed, totalPaid, paymentDate

Table: **vehicle**
VehicleID -> numOfSeats, colour, dailyRate, carModel, carLocation

Table: **booking**
BookingID -> VehicleID, client, description, startDate, startDate

7 - Normalization: 3NF

Booking

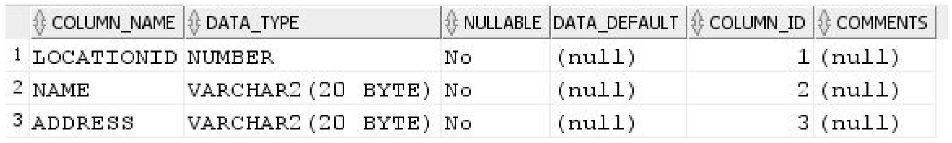


	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	BOOKINGID	NUMBER	No	(null)	1	(null)
2	TRANSACTIONID	NUMBER	Yes	(null)	2	(null)
3	VEHICLEID	NUMBER	Yes	(null)	3	(null)
4	CLIENT	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	DESCRIPTION	VARCHAR2(255 BYTE)	Yes	(null)	5	(null)
6	STARTDATE	DATE	No	(null)	6	(null)
7	ENDDATE	DATE	No	(null)	7	(null)

2NF – `TransactionID`, `vehicleID`, `client`, `description`, `startDate`, and `endDate` are all fully functionally dependent on the primary key `bookingID`.

3NF – `TransactionID`, `vehicleID`, `client`, `description`, `startDate`, and `endDate` are all functionally dependent on the primary key `bookingID`. They are all determined by `bookingID` and cannot be determined by another column.

Locations

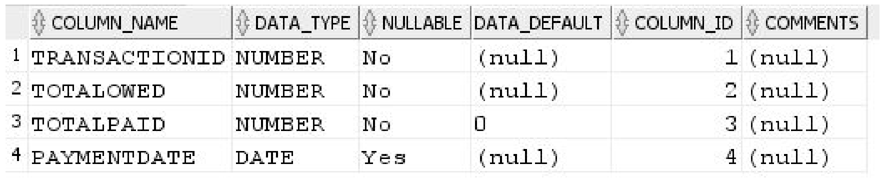


	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LOCATIONID	NUMBER	No	(null)	1	(null)
2	NAME	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	ADDRESS	VARCHAR2(20 BYTE)	No	(null)	3	(null)

2NF – `Name` and `address` are fully functionally dependent on the primary key `LocationID`.

3NF – `Name` and `address` of the location are functionally dependent on the primary key `LocationID`. They are all determined by `LocationID` and cannot be determined by another column.

Transactions

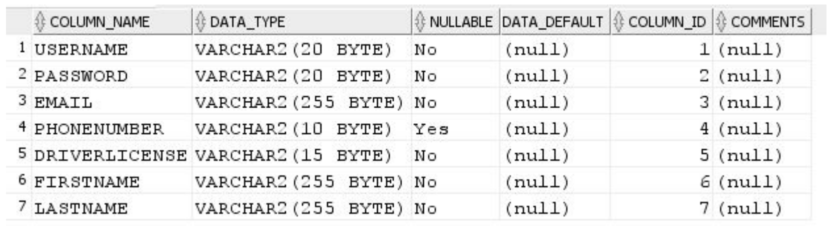


	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TRANSACTIONID	NUMBER	No	(null)	1	(null)
2	TOTALOWED	NUMBER	No	(null)	2	(null)
3	TOTALPAID	NUMBER	No	0	3	(null)
4	PAYMENTDATE	DATE	Yes	(null)	4	(null)

2NF – `TotalOwed`, `totalPaid`, and `paymentDate` are fully functionally dependent on the primary key `TransactionID`.

3NF – `TotalOwed`, `totalPaid`, and `paymentDate` are all functionally dependent on the primary key `TransactionID`. They are all determined by `TransactionID` and cannot be determined by another column.

UserAccount

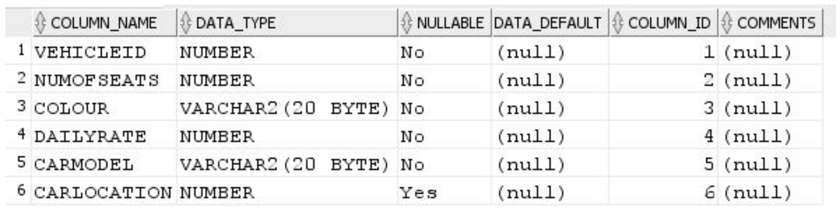


	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	USERNAME	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	PASSWORD	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	EMAIL	VARCHAR2(255 BYTE)	No	(null)	3	(null)
4	PHONENUMBER	VARCHAR2(10 BYTE)	Yes	(null)	4	(null)
5	DRIVERLICENSE	VARCHAR2(15 BYTE)	No	(null)	5	(null)
6	FIRSTNAME	VARCHAR2(255 BYTE)	No	(null)	6	(null)
7	LASTNAME	VARCHAR2(255 BYTE)	No	(null)	7	(null)

2NF – `Password`, `email`, `phoneNumber`, `driversLicense`, `firstName` and `lastName` are all fully functionally dependent on the primary key `username`.

3NF – `Password`, `email`, `phoneNumber`, `driversLicense`, `firstName` and `lastName` are all functionally dependent on the primary key `username`. They are all determined by `username` and cannot be determined by another column.

Vehicle



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	VEHICLEID	NUMBER	No	(null)	1	(null)
2	NUMOFSEATS	NUMBER	No	(null)	2	(null)
3	COLOUR	VARCHAR2(20 BYTE)	No	(null)	3	(null)
4	DAILYRATE	NUMBER	No	(null)	4	(null)
5	CARMODEL	VARCHAR2(20 BYTE)	No	(null)	5	(null)
6	CARLOCATION	NUMBER	Yes	(null)	6	(null)

2NF – `NumOfSeats`, `colour`, `dailyRate`, `carModel`, and `carLocation` are all fully functionally dependent on the primary key `VehicleID`.

3NF – `NumOfSeats`, `colour`, `dailyRate`, `carModel`, and `carLocation` are all functionally dependent on the primary key `VehicleID`. They are all determined by `VehicleID` and cannot be determined by another column.

8 - Normalization: 3NF/BCNF by Algorithm

Bernstein's Algorithm for Vehicle Table

Step 1: Find out facts about the real world, result is a list of attributes and FDs

Vehicles have a certain number of attributes that are relevant for a car rental system, such as:

- The number of seats for a car
- The car's model
- The car's colour

In a car rental system, other attributes are needed to run the business:

- Daily rental rate for each car
- Which location the car is currently stored in
- Vehicle ID to identify the car

Therefore, the functional dependency would be as follows:

```
VehicleID -> NumOfSeats, colour, dailyRate, carModel, carLocation
```

Step 2: Reduce the list of functional dependencies

The list of functional dependencies cannot be reduced any further.

Step 3: Find the keys

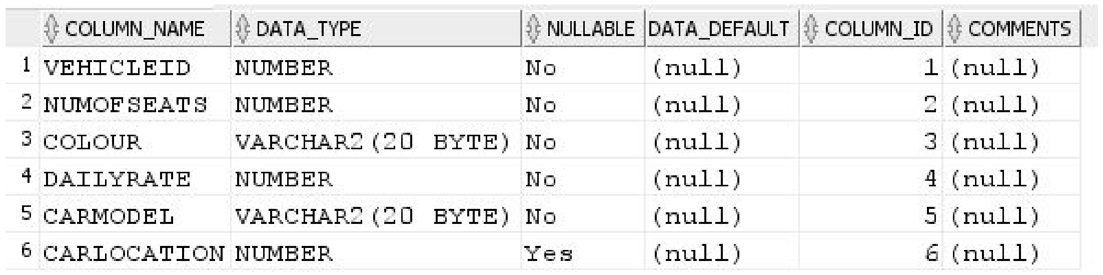
```
VehicleID -> NumOfSeats, colour, dailyRate, carModel, carLocation
```

Primary Key is `VehicleID`

Attributes/columns are `NumOfSeats`, `colour`, `dailyRate`, `carModel`, and `carLocation`

Step 4: Derive the final schema

Resulting schema:



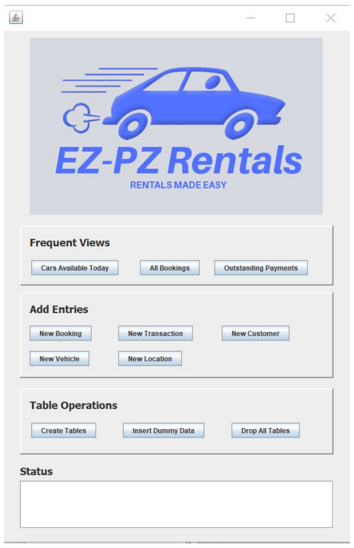
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	VEHICLEID	NUMBER	No	(null)	1	(null)
2	NUMOFSEATS	NUMBER	No	(null)	2	(null)
3	COLOUR	VARCHAR2(20 BYTE)	No	(null)	3	(null)
4	DAILYRATE	NUMBER	No	(null)	4	(null)
5	CARMODEL	VARCHAR2(20 BYTE)	No	(null)	5	(null)
6	CARLOCATION	NUMBER	Yes	(null)	6	(null)

9 - Final Remarks

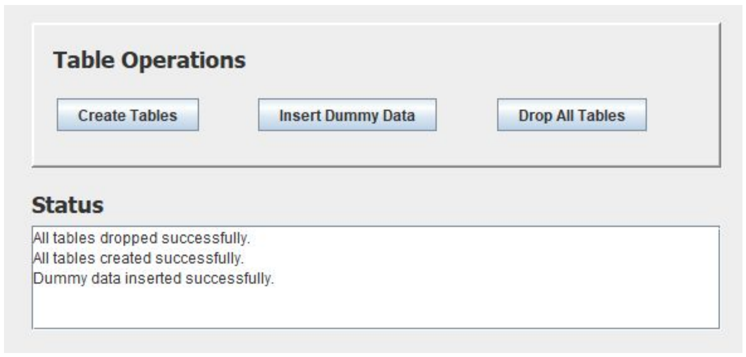
Overall this assignment has been very useful in teaching us how databases are created. Many times throughout the process, we had to backtrack and fix previous labs due to errors in the initial code. We took what was taught in class and applied it in our labs to create a useful database that can be used for a car rental company.

10 - Project Demo

Home page:

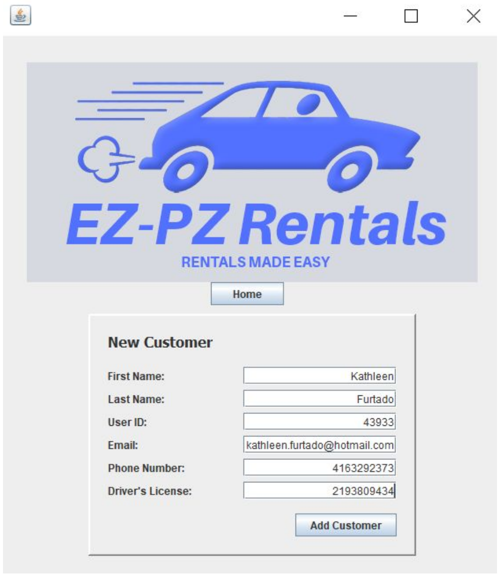


The program can create tables, insert dummy data, and drop all tables. Status is shown in the bottom text box:



```
All tables dropped successfully.
All tables created successfully.
Dummy data inserted successfully.
```

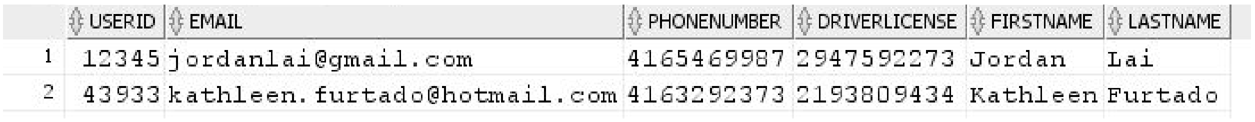
Adding a row (ex, adding a new customer):



New Customer

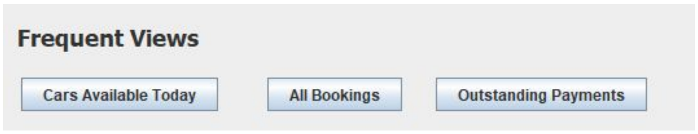
First Name:	Kathleen
Last Name:	Furtado
User ID:	43933
Email:	kathleen.furtado@hotmail.com
Phone Number:	4163292373
Driver's License:	2193809434

Successfully added:

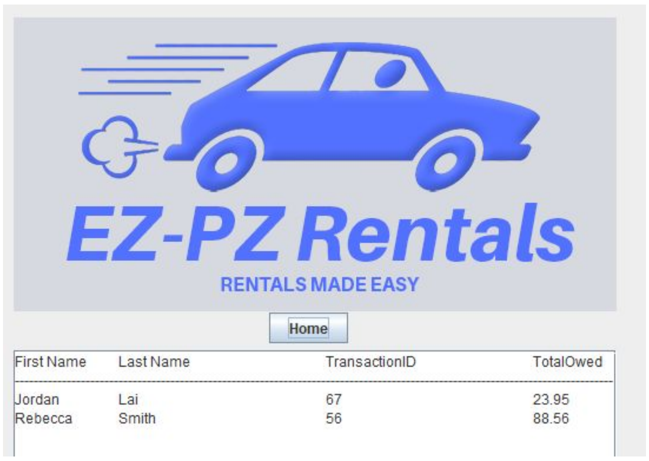


	USERID	EMAIL	PHONENUMBER	DRIVERLICENSE	FIRSTNAME	LASTNAME
1	12345	jordanlai@gmail.com	4165469987	2947592273	Jordan	Lai
2	43933	kathleen.furtado@hotmail.com	4163292373	2193809434	Kathleen	Furtado

Three main views can be displayed:



Displaying a view (ex, outstanding payments):



First Name	Last Name	TransactionID	TotalOwed
Jordan	Lai	67	23.95
Rebecca	Smith	56	88.56