Carbook Rentals

CARBOOK

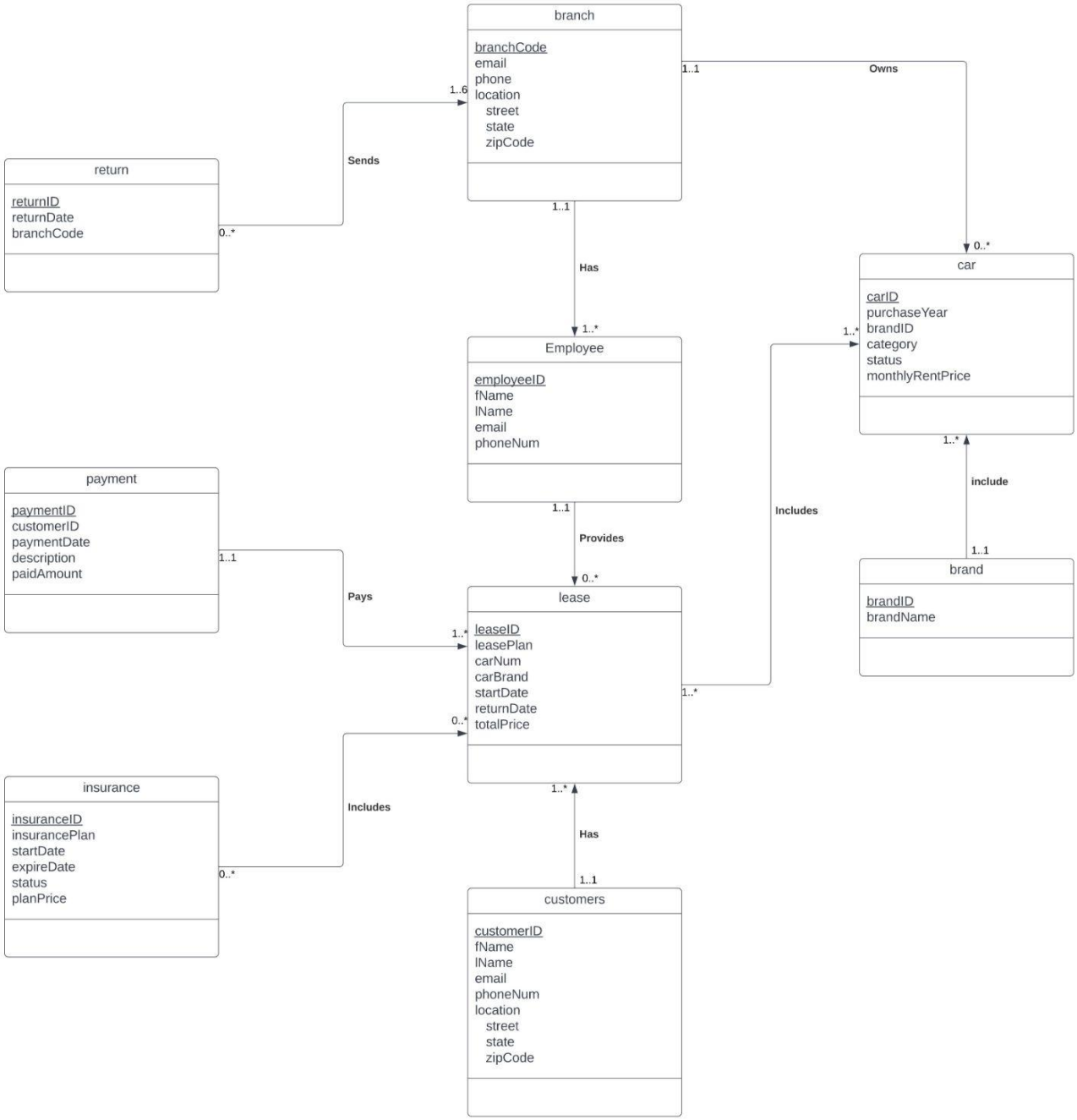
Business Scenario

A car rental service with a lot of locations and deals with a lot of information on a daily basis. With five branch locations, the company, Carbook, has a lot of cars that are either rented or leased for consumers to use. However not all locations have the same variety of cars that customers may desire.

The goal for our application is very simple. Our application allows for one location to see the availability of specific cars within the inventory of the whole company to ensure that they have enough cars in their location to supply for their upcoming orders. This application would also look into the orders processed in other locations every week to see if cars should be transferred among the different locations. In doing so, we hope that this would ensure that there is always a car that a consumer would want.

Normally when customers rent a car they want a very specific car and sometimes the car is unavailable for rent in a specific location. The key function can be defined as car available vs car unavailable (due to reasons such as checking). For example, a customer may want to rent a specific car from the location they are in. If the specific car isn't at that specific location, the employees can always look up other locations that may have the car. If the car is unavailable at all locations, we can then recommend other cars similar to the car the customer desired, either the same brand or category. Doing so allows the customer to stay with the company and keep customer loyalty. Allowing a location to possess a wide range of in-demand cars or something similar allows for the location to have a high chance of meeting the demands of their customers and risk losing business.

ER Model using UML Notation



Relationship sentences:

- branch and employee:

One **branch** may have one or more **employees**.
One **employee** must work for one and only one **branch**.

- branch and car:

One **branch** may own zero or more **cars**.
One **car** must stay at one and only one **branch**.

- return and branch

one **branch** would have zero or many **returns**.
one **return** record may belong to one and only one **branch**.

- brand and car:

One **brand** may include one or more **cars**.
One **car** must belong to one and only one **brand**.

- customers and lease:

One **customer** may have one or many **leases**.
One **lease** must be made by one and only one **customer**.

- employee and lease:

One **employee** could have zero or many **leases**.
One **lease** must come from one and only one **employee**.

- car and lease

one **car** may from one or many **leases**
one **lease** may have one or many **cars**

- payment and lease

one **payment** may from one or many **leases**
one **lease** must have and only one **payment**

- insurance and lease

one **insurance** may cover zero or many **leases**
one **lease** may have zero or many **insurance**

Converting the ERD to a Relational Model

RDM

1:*
branch(branchCode, email, phone, location)

Employee(employeeID, fName,IName, email, phoneNum, branchCode(FK))

brand(brandID, brandName)

car(carID, purchaseYear, brandID, category, status, monthlyRentPrice, branchCode(FK), brandID(FK))

customers(customerID, fName, IName, email, phoneNum, location)

payment(paymentID,customerID, paymentDate, description, paidAmount)

lease(leaseID,leasePlan, carNum, carBrand, startDate, returnDate, totalPrice, customerID(FK), paymentID(FK))

return(returnID, returnDate, branchCode, branchCode(FK))

:

insurance(insuranceID, insurancePlan, startDate, expireDate, status, planPrice)
lease(leaseID,leasePlan, carNum, carBrand, startDate, returnDate, totalPrice)
insurance_lease(insuranceID, leaseID)

lease(leaseID,leasePlan, carNum, carBrand, startDate, returnDate, totalPrice)
car(carID, purchaseYear, brandID, category, status, monthlyRentPrice)
lease_car(leaseID, carID)

Normalization:

branch:

Step 1: Key is branchCode
Step 2: Functional Dependencies

FD1: branchCode -> email, phone, location
FD2: email -> phone, location

Step 3:

key?	Yes, branchCode. In 1NF
Partial key dependency?	No, 2NF
Transitive dependency?	Yes, FD2

split:
R1 (email, phone, location)
R2 (branchCode, email)

No more, in 3NF

employee:

Step 1: Key is employeeID
Step 2: Functional Dependencies

FD1: employeeID -> fName, lName, email, phoneNum
FD2: email -> fName, lName, phoneNum

Step 3:

key?	Yes, employeeID. In 1NF
Partial key dependency?	No, 2NF
Transitive dependency?	Yes, FD2

split:

R1 (email, fName, lName, phoneNum)
R2 (employeeID, email)

Checked, no more, in 3NF

lease:

Step 1: Key is leaseID
Step 2: Functional Dependencies

FD1: leaseID -> leasePlan, carNum, carBrand, startDate, returnDate, totalPrice

Step 3:

key?	Yes, leaseID. In 1NF
Partial key dependency?	No, 2NF
Transitive dependency?	No

R1 (leaseID, leasePlan, carNum, carBrand, startDate, returnDate, totalPrice) in 3NF

customer:

Step 1: Key is customerID
Step 2: Functional Dependencies

FD1: customerID -> fName, lName, email, phoneNum, location
FD2: email -> fName, lName, phoneNum, location

Step 3:

key?	Yes, customerID. In 1NF
Partial key dependency?	No, 2NF
Transitive dependency?	Yes, FD2

split:

R1 (email, fName, lName, phoneNum, location)
R2 (customerID, email)

checked, no more. In 3NF

Car:

Step 1: Key is carID
Step 2: Functional Dependencies

FD1: carID -> purchaseYear, brandID, category, status, monthlyRentPrice

Step 3:

key?	Yes, carID. In 1NF
Partial key dependency?	No, in 2NF
Transitive dependency?	No

R1 (carID, purchaseYear, brandID, category, status, monthlyRentPrice) in 3NF

Brand:

Step 1: Key is brandID

Step 2: Functional Dependencies

FD1: brandID -> brandName

Step 3:

key?	Yes, brandID. In 1NF
Partial key dependency?	No, in 2NF
Transitive dependency?	No

R1 (brandID, brandName) in 3NF

return:

Step 1: Key is returnID

Step 2: Functional Dependencies

FD1: returnID -> returnDate, branchCode

Step 3:

key? Yes, returnID. In 1NF

Partial key dependency? No, 2NF

Transitive dependency? No

R1 (returnID, returnDate, branch Code) in 3NF

payment:

Step 1: Key is paymentID

Step 2: Functional Dependencies

FD1: paymentID -> customerID, paymentDate, description, paidAmount

Step 3:

key? Yes, paymentID. In 1NF

Partial key dependency? No, 2NF

Transitive dependency? No

R1 (paymentID, customerID, paymentDate, description, paidAmount) in 3NF

insurance:

Step 1: Key is insuranceID

Step 2: Functional Dependencies

FD1: insuranceID -> insurancePlan, startDate, expireDate, status, planPrice

Step 3:

key? Yes, insuranceID. In 1NF

Partial key dependency? No, 2NF

Transitive dependency? No

R1 (insuranceID, insurancePlan, startDate, expireDate, status, planPrice) in 3NF

Creating Tables with SQL

---branch:

```
CREATE TABLE branch(

branchCode VARCHAR(100) NOT NULL,
email VARCHAR(100),
phone NUMBER,
street VARCHAR(100),
state VARCHAR(2),
zipCode NUMBER,

CONSTRAINT pk_branch PRIMARY KEY (branchCode)

);

INSERT INTO branch VALUES
("bh01", "carbookBrooklyn@gmail.com", 6467458763, "Brooklyn","NY", 11207);

INSERT INTO branch VALUES
("bh02", "carbookQueens@yahoo.com", 7689584321,"Queens", "NY", 11356);

INSERT INTO branch VALUES
("bh03", "carbookBronx@gmail.com", 9178347633, "Bronx", "NY", 10452);

INSERT INTO branch VALUES
("bh04", "carbookManhattan@yahoo.com", 6378792341, "Manhattan", "NY", 10015);

INSERT INTO branch VALUES
```

("bh05", "carbookSI@gmail.com", 7123478911, "Staten Island", "NY", 10313);

branch

branchCode	email	phone	street	state	zipCode	Click to Add
bh01	carbookBrooklyn@gmail.com	6467458763	Brooklyn	NY	11207	
bh02	carbookQueens@yahoo.com	7689584321	Queens	NY	11356	
bh03	carbookBronx@gmail.com	9178347633	Bronx	NY	10452	
bh04	carbookManhattan@yahoo.com	6378792341	Manhattan	NY	10015	
bh05	carbookSI@gmail.com	7123478911	Staten Island	NY	10313	

---return:

CREATE TABLE return(

returnID VARCHAR(100) NOT NULL,
returnDate DATE,
branchCode VARCHAR(100) NOT NULL,

CONSTRAINT pk_return PRIMARY KEY (returnID),
CONSTRAINT fk_return FOREIGN KEY (branchCode) REFERENCES branch(branchCode)

);
INSERT INTO return VALUES("rd01", #05/02/2022#, "bh01");

INSERT INTO return VALUES("rd02", #7/3/2022#, "bh02");

INSERT INTO return VALUES("rd03", #12/15/2022#, "bh03");

INSERT INTO return VALUES("rd04", #09/23/2022#, "bh04");

INSERT INTO return VALUES("rd05", #03/26/2022#, "bh05");

return

returnID	returnDate	branchCode	Click to Add
rd01	5/2/2022	bh01	
rd02	7/3/2022	bh02	
rd03	12/15/2022	bh03	
rd04	9/23/2022	bh04	
rd05	3/26/2022	bh05	

---employee:

```
CREATE TABLE employee(

employeeID VARCHAR(100) NOT NULL,
fName VARCHAR(100),
lName VARCHAR(100),
email VARCHAR(100),
phoneNum NUMBER,
branchCode VARCHAR(100),

CONSTRAINT pk_employee PRIMARY KEY (employeeID),
CONSTRAINT fk_employee FOREIGN KEY (branchCode) REFERENCES
branch(branchCode)

);


INSERT INTO employee VALUES
("e001","Jessica","Gonzalez","Jessica2000@gmail.com",9297718752,"bh01")

INSERT INTO employee VALUES
("e002","Jacob","Morris","Jacob1999@yahoo.com",9299871254,"bh02")

INSERT INTO employee VALUES
("e003","Melissa","Kelly","MKL996@gmail.com",9290241234,"bh03")

INSERT INTO employee VALUES
("e004","Stephen","Chavez","ST567@outlook.com",3145768989,"bh04")

INSERT INTO employee VALUES
("e005","Shannon","Patel","PPH358@yahoo.com",6145892323,"bh05")
```

employee

employeeID	fName	lName	email	phoneNum	branchCode	Click to Add
e001	Jessica	Gonzalez	Jessica2000@gmail.com	9297718752	bh01	
e002	Jacob	Morris	Jacob1999@yahoo.com	9299871254	bh02	
e003	Melissa	Kelly	MKL996@gmail.com	9290241234	bh03	
e004	Stephen	Chavez	ST567@outlook.com	3145768989	bh04	
e005	Shannon	Patel	PPH358@yahoo.com	6145892323	bh05	
e006	Olivia	Green	OlivaG@gmail.com	3478762323	bh01	
e007	Lucas	Foster	LF1997@yahoo.com	9296448679	bh03	
e008	Henry	Ross	henry.ross@outlook.com	3476648903	bh03	
e009	Sophia	Reyes	SophiaRe11@gmail.com	3187856464	bh04	
e010	Amelia	Torres	TorresA3@yahoo.com	9293476702	bh05	
e011	Noah	Ho	NH.70@gmail.com	7183588899	bh01	
e012	Brian	Flores	Brain.Flores20@yahoo.com	8706333953	bh05	
e013	Kevin	Lee	KLEE1999@gmail.com	9176744706	bh02	
e014	Laura	Hernandez	LaHe@gmail.com	5512678036	bh04	
e015	Sarah	Taylor	SarahT16@yahoo.com	7164561800	bh02	

---brand:

```
CREATE TABLE brand(

brandID VARCHAR(100) NOT NULL,
brandName VARCHAR(100),

CONSTRAINT pk_brand PRIMARY KEY (brandID)

);

INSERT INTO brand VALUES ("bd01","Toyota")
INSERT INTO brand VALUES ("bd02","Volvo")
INSERT INTO brand VALUES ("bd03","BMW")
INSERT INTO brand VALUES ("bd04","Ford")
INSERT INTO brand VALUES ("bd05","GMC")
INSERT INTO brand VALUES ("bd06","Dodge")
INSERT INTO brand VALUES ("bd07","Nissan")
```

brand

brandID	brandName	Click to Add
bd01	Toyota	
bd02	Volvo	
bd03	BMW	
bd04	Ford	
bd05	GMC	
bd06	Dodge	
bd07	Nissan	

---car:

```
CREATE TABLE car(

carID VARCHAR(100) NOT NULL,
purchaseYear NUMBER,
category VARCHAR(100),
status VARCHAR(100),
monthlyRentPrice NUMBER,
branchCode VARCHAR(100),
brandID VARCHAR(100),

CONSTRAINT pk_car PRIMARY KEY (carID),
CONSTRAINT fk_car FOREIGN KEY (branchCode) REFERENCES branch(branchCode),
CONSTRAINT fk2_car FOREIGN KEY(brandID) REFERENCES brand(brandID)

);

INSERT INTO car VALUES ("ca001",2018,"suv","available",500,"bh02","bd01")

INSERT INTO car VALUES ("ca002",2020,"truck","available",700,"bh05","bd05")

INSERT INTO car VALUES ("ca003",2021,"sedan","available",300,"bh01","bd02")

INSERT INTO car VALUES ("ca004",2019,"wagon","available",550,"bh03","bd04")

INSERT INTO car VALUES ("ca005",2018,"sedan","available",270,"bh04","bd07")
```

car

carID	purchaseYear	category	status	monthlyRentPrice	branchCode	brandID	Click to Add
ca001	2018	suv	available	500	bh02	bd01	
ca002	2020	truck	available	700	bh05	bd05	
ca003	2021	sedan	available	300	bh01	bd02	
ca004	2019	wagon	available	550	bh03	bd04	
ca005	2018	sedan	available	270	bh04	bd07	
ca006	2019	suv	available	550	bh04	bd01	
ca007	2017	sedan	checking	250	bh02	bd07	
ca008	2020	wagon	available	600	bh03	bd03	
ca009	2021	suv	available	620	bh02	bd02	
ca010	2018	truck	available	640	bh01	bd05	
ca011	2018	sedan	available	260	bh04	bd01	
ca012	2019	sedan	checking	300	bh05	bd04	
ca013	2017	truck	available	580	bh02	bd05	
ca014	2020	suv	available	570	bh02	bd06	
ca015	2020	wagon	available	570	bh01	bd07	
ca016	2019	truck	available	680	bh05	bd07	
ca017	2021	suv	available	625	bh04	bd01	
ca018	2018	suv	available	510	bh03	bd03	
ca019	2017	sedan	available	250	bh04	bd01	
ca020	2019	wagon	available	555	bh03	bd04	
ca021	2020	truck	available	690	bh05	bd03	
ca022	2018	suv	available	520	bh01	bd07	
ca023	2017	sedan	available	260	bh03	bd04	
ca024	2019	sedan	available	300	bh04	bd02	
ca025	2019	suv	available	530	bh05	bd06	
ca026	2020	wagon	available	570	bh02	bd03	
ca027	2021	sedan	available	360	bh01	bd06	
ca028	2018	wagon	available	510	bh05	bd05	
ca029	2019	truck	available	690	bh03	bd02	
ca030	2017	suv	available	490	bh02	bd06	

---payment:

CREATE TABLE payment (

paymentID VARCHAR(100) NOT NULL,
customerID VARCHAR(100),
paymentDate DATE,
description VARCHAR(100),
paidAmount NUMBER,

CONSTRAINT pk_payment PRIMARY KEY (paymentID)

);

INSERT INTO payment VALUES ("p0001","cu0001",#1/3/2020#,"none",1800)

INSERT INTO payment VALUES ("p0002","cu0010",#5/16/2019#,"none",1650)

INSERT INTO payment VALUES ("p0003","cu0008",#7/7/2022#,"1/4 of payment are deducted",540)

INSERT INTO payment VALUES ("p0004","cu0006",#8/12/2019#,"employee discount - 15% off",600)

INSERT INTO payment VALUES ("p0005","cu0002",#4/26/2020#,"none",810)

INSERT INTO payment VALUES ("p0006","cu0005",#2/1/2020#,"none",910)

payment

paymentID	customerID	paymentDate	description	paidAmount	Click to Add
p0001	cu0001	1/3/2020	none	1800	
p0002	cu0010	5/16/2019	none	1650	
p0003	cu0008	7/7/2022	none	540	
p0004	cu0006	8/12/2019	none	600	
p0005	cu0002	4/26/2020	none	810	
p0006	cu0005	2/1/2020	none	910	
p0007	Cu0011	3/7/2021	none	1350	
p0008	Cu0017	8/5/2019	none	500	
p0009	Cu0019	12/1/2019	none	1140	
p0010	Cu0012	10/14/2020	none	500	
p0011	Cu0014	8/24/2021	used gift card, $140 was deducted	2800	
p0012	Cu0018	1/4/2022	none	555	
p0013	Cu0016	4/18/2022	used gift card, $100 was deducted	520	
p0014	Cu0007	9/22/2020	10% off due to the anniversary celebration	936	
p0015	Cu0009	6/19/2019	none	1710	
p0016	Cu0013	4/21/2018	none	690	
p0017	Cu0020	7/7/2021	50% off due to the investor discount	960	
p0018	Cu0015	9/12/2020	none	510	

<u>---customers:</u>

CREATE TABLE customers (

customerID VARCHAR(100) NOT NULL,
fName VARCHAR(100),
IName VARCHAR(100),
email VARCHAR(100),
phoneNum NUMBER,
street VARCHAR(100),
state VARCHAR(100),
zipCode NUMBER,

CONSTRAINT pk_customers PRIMARY KEY (customerID)

);

INSERT INTO customers VALUES ('Cu0001', 'Liu', 'Meng', 'Linlily1234@yahoo.com', 9118342543, '280 Regis Dr, Staten Island', 'NY', 10314);

INSERT INTO customers VALUES

('Cu0002', 'Wu', 'Jenny', 'Jennywuu@gmail.com', 7186329087, '151-15 85th Dr, Queens', 'NY', 11432);

INSERT INTO customers VALUES
('Cu0003', 'Billy', 'Forbs', 'Billlyyuu21@gmail.com', 6457869021, '856 Quincy St, Brooklyn', 'NY', 11221);

INSERT INTO customers VALUES
('Cu0004', 'Alex', 'Novak', 'Alexxx8910@gmail.com', 7328904534, '12 W 12th St, New York', 'NY', 10179);

INSERT INTO customers VALUES
('Cu0005', 'Lily', 'Connor', 'Lillyjeffing@outllook.com', 9348723452, '260 Pleasant Ave, New York', 'NY', 10180);

customers

customerID	fName	lName	email	phoneNum	street	state	zipCode	Click
Cu0001	Liu	Meng	Linlily1234@yahoo.com	9118342543	280 Regis Dr, Staten Island	NY	10314	
Cu0002	Wu	Jenny	Jennywuu@gmail.com	7186329087	151-15 85th Dr, Queens	NY	11432	
Cu0003	Billy	Forbs	Billlyyuu21@gmail.com	6457869021	856 Quincy St, Brooklyn	NY	11221	
Cu0004	Alex	Novak	Alexxx8910@gmail.com	7328904534	12 W 12th St, New York	NY	10179	
Cu0005	Lily	Connor	Lillyjeffing@outllook.com	9348723452	260 Pleasant Ave, New York	NY	10180	
Cu0006	Dennis	Galloway	Dennisss16@gmail.com	6179054761	695 Park Ave, New York	NY	10118	
Cu0007	Sandra	Bray	BrayS888@yahoo.com	7145082453	291 W 231st St, Bronx	NY	10463	
Cu0008	Terry	Nieves	Terrr909@outlook.com	9297814653	1197 Sutter Ave. at, Crystal St, Brooklyn	NY	11208	
Cu0009	Clayton	Mcgrath	Claymc@gmail.com	4143579522	155-06 Roosevelt Ave, Queens	NY	11345	
Cu0010	Patrick	Kaufman	Patrickkau8@gmail.com	3475689421	476 5th Ave, New York	NY	10018	
Cu0011	Sarah	Holden	DrSarahH@yahoo.com	3547802525	21 Robin Rd, Staten Island	NY	10305	
Cu0012	Carol	Krause	Carol.Kr16@outlook.com	2976392083	200 Clarke Ave, Staten Island	NY	10306	
Cu0013	Nelson	Braun	Nelson1994@gmail.com	9297856824	310 E Kingsbridge Rd, Bronx	NY	10458	
Cu0014	Connie	Werner	Conniewww@yahoo.com	3145695354	135th St and, Malcolm X Blvd, New York	NY	10037	
Cu0015	Eileen	Vera	VeraEil165@gmail.com	7180613232	910 Morris Ave, Bronx	NY	10451	
Cu0016	Ben	Davies	Ben16Da@yahoo.com	7187542451	87 Crosby St, New York	NY	10012	
Cu0017	Roger	Saenz	Roger2001@gmail.com	9293228746	2147 Barnes Ave, Bronx	NY	10462	
Cu0018	Ella	Forbes	EllaFo@gmail.com	3145609362	5 Central Ave, Staten Island	NY	10301	
Cu0019	Tiffany	Riddle	Tiffanyyy012@yahoo.com	6140346924	41-17 Main St, Flushing	NY	11355	
Cu0020	Tara	Rosa	Tara.Rosa@outlook.com	7010623758	221 E 71st St, New York	NY	10021	

---insurance:

CREATE TABLE insurance (

insuranceID VARCHAR(100) NOT NULL,
insurancePlan VARCHAR(100),
startDate DATE,
expireDate DATE,
status VARCHAR(100),
plainPrice NUMBER,

CONSTRAINT pk_insurance PRIMARY KEY (insuranceID)

);

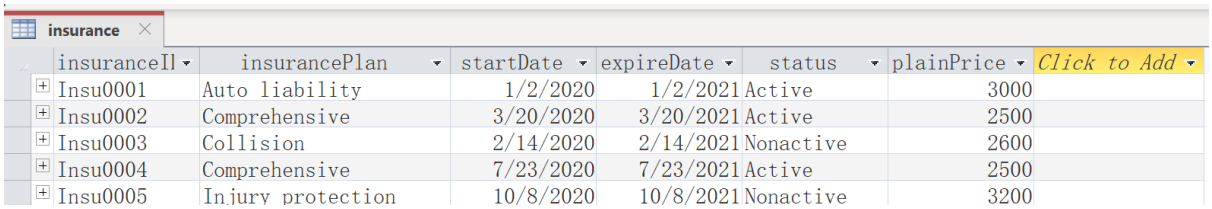
INSERT INTO insurance VALUES
('Insu0001', 'Auto liability', #1/2/2020#, #1/2/2021#, 'Active', 3000);

INSERT INTO insurance VALUES
('Insu0002', 'Comprehensive', #3/20/2020#, #3/20/2021#, 'Active', 2500);

INSERT INTO insurance VALUES
('Insu0003', 'Collision', #2/14/2020#, #2/14/2021#, 'Nonactive', 2600);

INSERT INTO insurance VALUES
('Insu0004', 'Comprehensive', #7/23/2020#, #7/23/2021#, 'Active', 2500);

INSERT INTO insurance VALUES
('Insu0005', 'Injury protection', #10/8/2020#, #10/8/2021#, 'Nonactive', 3200);



insurance

insuranceII	insurancePlan	startDate	expireDate	status	plainPrice	Click to Add
Insu0001	Auto liability	1/2/2020	1/2/2021	Active	3000	
Insu0002	Comprehensive	3/20/2020	3/20/2021	Active	2500	
Insu0003	Collision	2/14/2020	2/14/2021	Nonactive	2600	
Insu0004	Comprehensive	7/23/2020	7/23/2021	Active	2500	
Insu0005	Injury protection	10/8/2020	10/8/2021	Nonactive	3200	

---lease:

CREATE TABLE lease (

leaseID VARCHAR(100) NOT NULL,
leasePlan VARCHAR(100),
carNum NUMBER,
carBrand VARCHAR(100),
startDate DATE,
returnDate DATE,
totalPrice NUMBER,

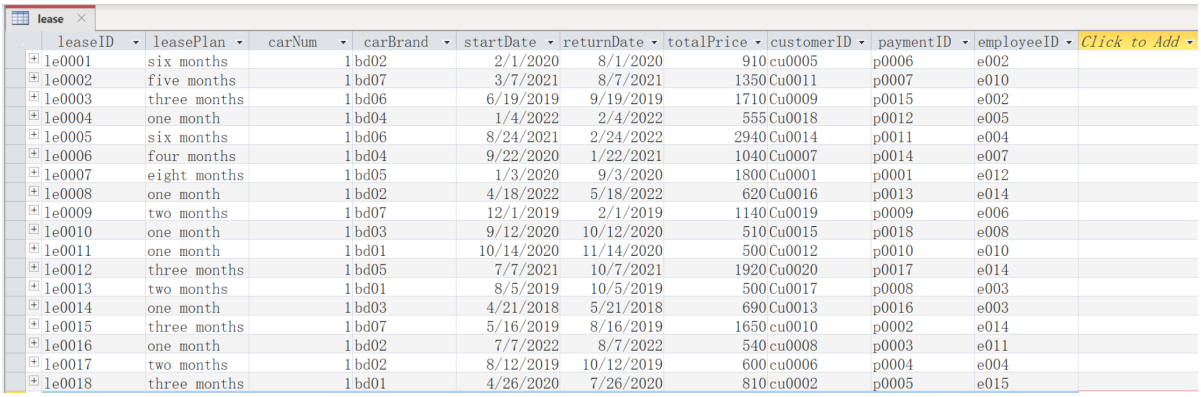
customerID VARCHAR(100),

paymentID VARCHAR(100),
employeeID VARCHAR(100),

CONSTRAINT pk_lease PRIMARY KEY (leaseID),
CONSTRAINT fk_lease FOREIGN KEY (customerID) REFERENCES customers(customerID),
CONSTRAINT fk2_lease FOREIGN KEY (paymentID) REFERENCES payment(paymentID),
CONSTRAINT fk3_lease FOREIGN KEY (employeeID) REFERENCES
employee(employeeID)

);

INSERT INTO lease VALUES ('le0001', 'six months', 1, 'bd02', #2/1/2020#, #8/1/2020#, 910,
'cu0005','p0006','e002')



lease

leaseID	leasePlan	carNum	carBrand	startDate	returnDate	totalPrice	customerID	paymentID	employeeID	Click to Add
le0001	six months	1	bd02	2/1/2020	8/1/2020	910	cu0005	p0006	e002	
le0002	five months	1	bd07	3/7/2021	8/7/2021	1350	Cu0011	p0007	e010	
le0003	three months	1	bd06	6/19/2019	9/19/2019	1710	Cu0009	p0015	e002	
le0004	one month	1	bd04	1/4/2022	2/4/2022	555	Cu0018	p0012	e005	
le0005	six months	1	bd06	8/24/2021	2/24/2022	2940	Cu0014	p0011	e004	
le0006	four months	1	bd04	9/22/2020	1/22/2021	1040	Cu0007	p0014	e007	
le0007	eight months	1	bd05	1/3/2020	9/3/2020	1800	Cu0001	p0001	e012	
le0008	one month	1	bd02	4/18/2022	5/18/2022	620	Cu0016	p0013	e014	
le0009	two months	1	bd07	12/1/2019	2/1/2019	1140	Cu0019	p0009	e006	
le0010	one month	1	bd03	9/12/2020	10/12/2020	510	Cu0015	p0018	e008	
le0011	one month	1	bd01	10/14/2020	11/14/2020	500	Cu0012	p0010	e010	
le0012	three months	1	bd05	7/7/2021	10/7/2021	1920	Cu0020	p0017	e014	
le0013	two months	1	bd01	8/5/2019	10/5/2019	500	Cu0017	p0008	e003	
le0014	one month	1	bd03	4/21/2018	5/21/2018	690	Cu0013	p0016	e003	
le0015	three months	1	bd07	5/16/2019	8/16/2019	1650	cu0010	p0002	e014	
le0016	one month	1	bd02	7/7/2022	8/7/2022	540	cu0008	p0003	e011	
le0017	two months	1	bd02	8/12/2019	10/12/2019	600	cu0006	p0004	e004	
le0018	three months	1	bd01	4/26/2020	7/26/2020	810	cu0002	p0005	e015	

Many-To-Many relationships:

—insurance-lease Relation:

CREATE TABLE LeaseInsuranceRelation(

leaseID VARCHAR(100) NOT NULL,
insuranceID VARCHAR(100) NOT NULL,

```
CONSTRAINT fk_LeaseInsuranceRelation FOREIGN KEY (leaseID) REFERENCES
lease(leaseID),
CONSTRAINT fk2_LeaseInsuranceRelation FOREIGN KEY (insuranceID) REFERENCES
insurance(insuranceID)

);
```

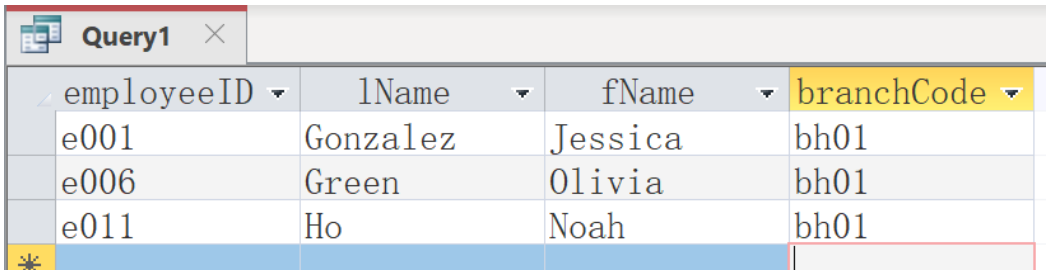
—lease-car Relation:

```
CREATE TABLE LeaseCarRelation(

leaseID VARCHAR(100) NOT NULL,
carID VARCHAR(100) NOT NULL,

CONSTRAINT fk_LeaseCarRelation FOREIGN KEY (leaseID) REFERENCES lease(leaseID),
CONSTRAINT fk2_LeaseCarRelation FOREIGN KEY (carID) REFERENCES car(carID)

);
```

Senario Part:

1. Write a query to display the employeeID, last name, first name, and branchCode for all employees who work in the brooklyn branch store.

```
SELECT employeeID, lName, fName, branchCode
FROM employee
Where branchCode = 'bh01'
```



employeeID	lName	fName	branchCode
e001	Gonzalez	Jessica	bh01
e006	Green	Olivia	bh01
e011	Ho	Noah	bh01

2. Write a query to display carID, purchaseYear, category and rent price for all cars whose monthly rent price is between $400 and $600 in ascending order by price.

```
SELECT carID, purchaseYear, category, monthlyRentPrice
FROM car
WHERE monthlyRentPrice BETWEEN 400 and 600
ORDER BY monthlyRentPrice ASC
```

Query1

carID	purchaseYear	category	monthlyRentPrice
ca030	2017	suv	490
ca001	2018	suv	500
ca028	2018	wagon	510
ca018	2018	suv	510
ca022	2018	suv	520
ca025	2019	suv	530
ca006	2019	suv	550
ca004	2019	wagon	550
ca020	2019	wagon	555
ca026	2020	wagon	570
ca015	2020	wagon	570
ca014	2020	suv	570
ca013	2017	truck	580
ca008	2020	wagon	600

3. Write a query to display insuranceID, insurance Plan, expire Date and plna Price for insurances which will expire before 7/1/2021, then sort the results in descending order by planPrice.

```
SELECT insuranceID, insurancePlan, expireDate, plainPrice
FROM insurance
WHERE expireDate < #7/1/2021#
ORDER BY plainPrice DESC
```

Query1

insuranceII	insurancePlan	expireDate	plainPrice
Insu0001	Auto liability	1/2/2021	3000
Insu0003	Collision	2/14/2021	2600
Insu0002	Comprehensive	3/20/2021	2500

4. Write a query to display the CarID, category, monthlyRentPrice, branchCode, branch location, brand name for all cars with price $570.

```
SELECT c.carID, category, monthlyRentPrice, bh.branchCode, street, bd.brandName
FROM car c, branch bh, brand bd
WHERE c.branchCode = bh.branchCode AND c.brandID = bd.brandID
AND monthlyRentPrice = 570
```

Query1

carID	category	monthlyRentPrice	branchCode	street	brandName
ca014	suv	570	bh02	Queens	Dodge
ca015	wagon	570	bh01	Brooklyn	Nissan
ca026	wagon	570	bh02	Queens	BMW

5. Write a query to display leaseID, leasePlan, carBrand, totalPrice in lease table, with lease plan of 1 month, and the rental cars with brand of Volvo (bd02).

```
Select leaseID, leasePlan, carBrand, totalPrice
From lease
Where leasePlan Like "one*" AND carBrand = "bd02"
```

Query1

leaseID	leasePlan	carBrand	totalPrice
le0008	one month	bd02	620
le0016	one month	bd02	540
*			

6. Write a query to display the leaseID, leasePlan, returnDate, paymentID, totalPrice for all leases with return date after 1/1/2020 and sort the results in ascending order by total Price.

```
SELECT leaseID, leasePlan, returnDate, paymentID, totalPrice
FROM lease
WHERE returnDate >= #1/1/2020#
ORDER BY totalPrice ASC
```

Query1

leaseID	leasePlan	returnDate	paymentID	totalPrice
le0011	one month	11/14/2020	p0010	500
le0010	one month	10/12/2020	p0018	510
le0016	one month	8/7/2022	p0003	540
le0004	one month	2/4/2022	p0012	555
le0008	one month	5/18/2022	p0013	620
le0018	three months	7/26/2020	p0005	810
le0001	six months	8/1/2020	p0006	910
le0006	four months	1/22/2021	p0014	1040
le0002	five months	8/7/2021	p0007	1350
le0007	eight months	9/3/2020	p0001	1800
le0012	three months	10/7/2021	p0017	1920
le0005	six months	2/24/2022	p0011	2940

7. Write a query to display employeeID, first name, last name, email, carBrand for employee who make up leasePlan for customer who want to rent car with brand "Nissan" (bd07).

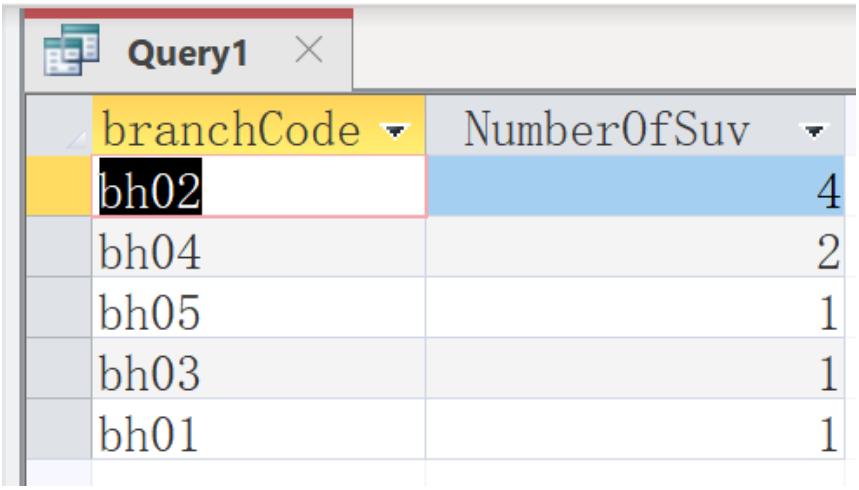
```
SELECT e.employeeID, fName, lName, email, l.carBrand
FROM employee e, lease l
WHERE e.employeeID = l.employeeID
AND l.carBrand = "bd07"
```

Query1

employeeID	fName	lName	email	carBrand
e010	Amelia	Torres	TorresA3@yahoo.com	bd07
e006	Olivia	Green	OlivaG@gmail.com	bd07
e014	Laura	Hernandez	LaHe@gmail.com	bd07

8. Write a query to display branchCode,category for how many suv cars in each branch (in descending order).

```
SELECT branchCode,COUNT(category) AS NumberOfSuv
FROM car
WHERE category = "suv"
GROUP BY branchCode
ORDER BY COUNT(category) DESC
```

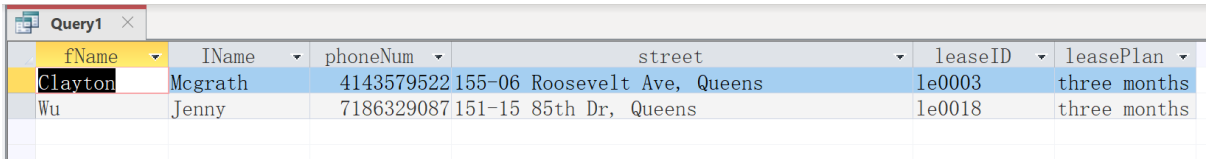


Query1

branchCode	NumberOfSuv
bh02	4
bh04	2
bh05	1
bh03	1
bh01	1

9. Write a query to display firstname, lastName, phoneNumber, street address, leaseID, leasePlan for how many customers live in Queens and make 3 months lease plan, then sort by leaseID

```
Select c.fName, lName, phoneNum, street, l.leaseID, leasePlan
From customers c, lease l
Where c.customerID = l.customerID
AND l.leasePlan = "three months" AND (c.street LIKE "*Queens" OR c.street LIKE
"*Flushing")
Order by l.leaseID
```



Query1

fName	lName	phoneNum	street	leaseID	leasePlan
Clayton	Mcgrath	4143579522	155-06 Roosevelt Ave, Queens	le0003	three months
Wu	Jenny	7186329087	151-15 85th Dr, Queens	le0018	three months

10. Write a query to display paymentID, paymentDate, description and paidAmount for all payment have paymentDate between 1/1/2020 and 11/1/2020.

```
SELECT paymentID, paymentDate, description, paidAmount
FROM payment
WHERE paymentDate BETWEEN #1/1/2020# AND #11/1/2020#
```

Query1

paymentID	paymentDate	description	paidAmount
p0001	1/3/2020	none	1800
p0005	4/26/2020	none	810
p0006	2/1/2020	none	910
p0010	10/14/2020	none	500
p0014	9/22/2020	10% off due to the anniversary celebration	936
p0018	9/12/2020	none	510

Conclusion:

Through the course of this project, we have learned how to build a database application through following the appropriate steps. The steps that we found most difficult was developing the ER diagram and normalizing it. Creating the SQL Tables and writing queries for the scenario part was most time consuming. We found that having a good diagram that fully demonstrates how different elements of the database are related is an important element of building a database application.

If we were to do it again, we would focus on spending more time building the ER Diagram so that we could integrate more entities and build more sophisticated relationships that create a more feature-packed application that can support all the needs of the rental company, such as keeping track of accidents, repairs and more of the day-to-day operational aspects of the car rental business. Overall however, our database application is able to support a car rental business comprehensively and operate the most important elements of their business efficiently. We have learned a lot of practical skills through this experience and have thoroughly enjoyed this project.