Uber Database

Andrew Arrigo

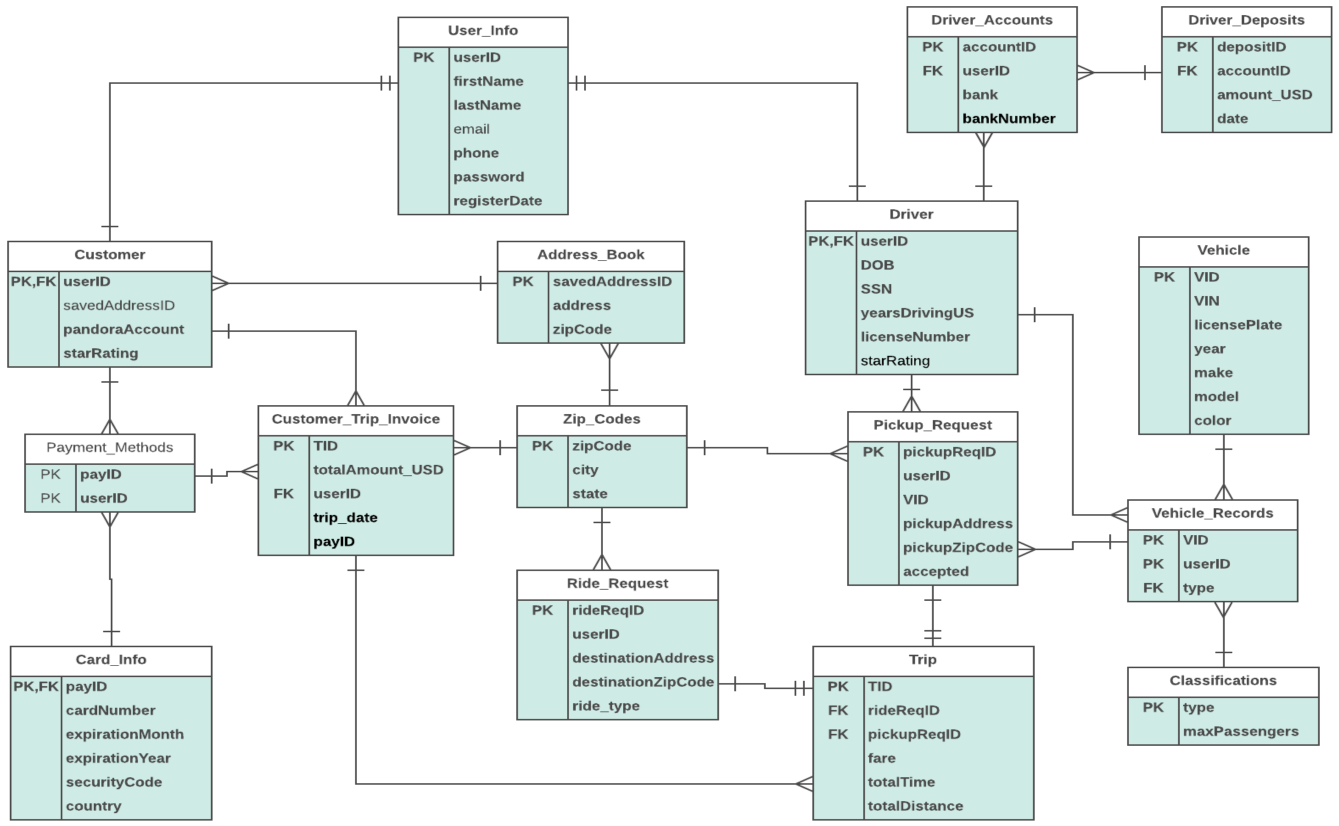
Table of Contents

Executive Summary

Within the past several years, the world of transportation has been taken over drastically by Uber. Uber is a transportation company that allows for people to become a driver, passenger (customer), or both driver/passenger. As Uber’s customer and driver rate has significantly increased, the need for a well designed database is crucial.

The database created will show how passenger information is stored in the database in comparison to driver information.

E/R Diagram



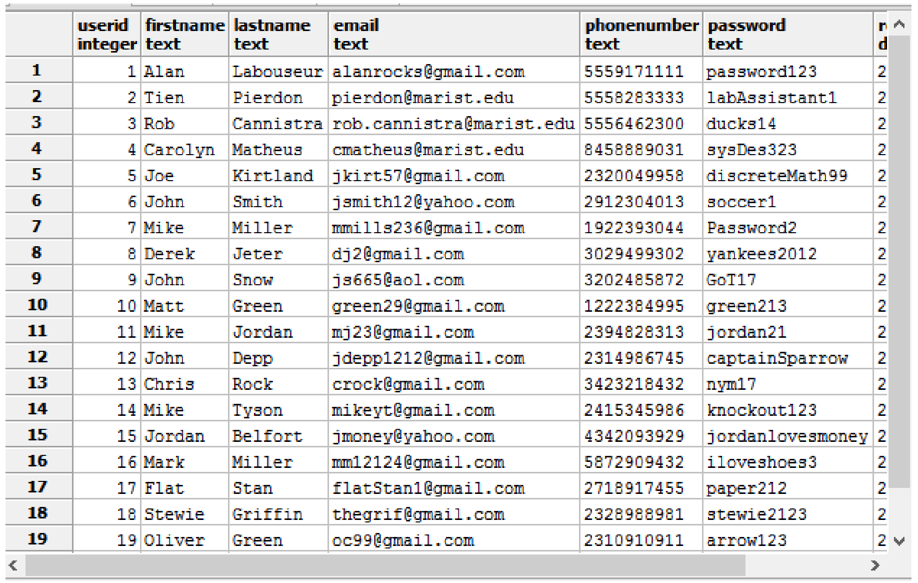
User_Info
PK userID
firstName
lastName
email
phone
password
registerDate
Driver_Accounts
PK accountID
FK userID
bank
bankNumber
Driver_Deposits
PK depositID
FK accountID
amount_USD
date
Customer
PK,FK userID
savedAddressID
pandoraAccount
starRating
Address_Book
PK savedAddressID
address
zipCode
Driver
PK,FK userID
DOB
SSN
yearsDrivingUS
licenseNumber
starRating
Vehicle
PK VID
VIN
licensePlate
year
make
model
color
Payment_Methods
PK payID
PK userID
Customer_Trip_Invoice
PK TID
totalAmount_USD
FK userID
trip_date
payID
Zip_Codes
PK zipCode
city
state
Pickup_Request
PK pickupReqID
userID
VID
pickupAddress
pickupZipCode
accepted
Vehicle_Records
PK VID
PK userID
FK type
Ride_Request
PK rideReqID
userID
destinationAddress
destinationZipCode
ride_type
Card_Info
PK,FK payID
cardNumber
expirationMonth
expirationYear
securityCode
country
Trip
PK TID
FK rideReqID
FK pickupReqID
fare
totalTime
totalDistance
Classifications
PK type
maxPassengers

User_Info Table

Functional Dependencies: userID → firstName, lastName, email, phoneNumber, password, registerDate

Displays account information about users signed up with Uber accounts

```
CREATE TABLE User_Info(
userID int NOT NULL,
firstName text NOT NULL,
lastName text NOT NULL,
email text NOT NULL,
phoneNumber text NOT NULL,
password text NOT NULL,
registerDate date NOT NULL,
PRIMARY KEY (userID)
)
;
```



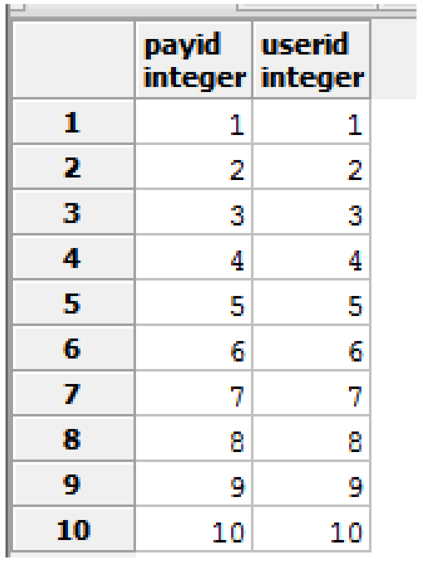
	userid	firstname	lastname	email	phonenumber	password	r
	integer	text	text	text	text	text	d
1	1	Alan	Labouseur	alanrocks@gmail.com	5559171111	password123	2
2	2	Tien	Pierdon	pierdon@marist.edu	5558283333	labAssistant1	2
3	3	Rob	Cannistra	rob.cannistra@marist.edu	5556462300	ducks14	2
4	4	Carolyn	Matheus	cmatheus@marist.edu	8458889031	sysDes323	2
5	5	Joe	Kirtland	jkirt57@gmail.com	2320049958	discreteMath99	2
6	6	John	Smith	jsmith12@yahoo.com	2912304013	soccer1	2
7	7	Mike	Miller	mmills236@gmail.com	1922393044	Password2	2
8	8	Derek	Jeter	dj2@gmail.com	3029499302	yankees2012	2
9	9	John	Snow	js665@aol.com	3202485872	GoT17	2
10	10	Matt	Green	green29@gmail.com	1222384995	green213	2
11	11	Mike	Jordan	mj23@gmail.com	2394828313	jordan21	2
12	12	John	Depp	jdepp1212@gmail.com	2314986745	captainSparrow	2
13	13	Chris	Rock	crock@gmail.com	3423218432	nym17	2
14	14	Mike	Tyson	mikeyt@gmail.com	2415345986	knockout123	2
15	15	Jordan	Belfort	jmoney@yahoo.com	4342093929	jordanlovesmoney	2
16	16	Mark	Miller	mm12124@gmail.com	5872909432	iloveshoes3	2
17	17	Flat	Stan	flatStan1@gmail.com	2718917455	paper212	2
18	18	Stewie	Griffin	thegrif@gmail.com	2328988981	stewie2123	2
19	19	Oliver	Green	oc99@gmail.com	2310910911	arrow123	2

Payment_Methods Table

Displays users payID that coincides with each userID

```
CREATE TABLE Payment_Methods(
payID int NOT NULL,
userID int NOT NULL,
PRIMARY KEY (payID)
)
;
```

Functional Dependencies:
payID, userID →



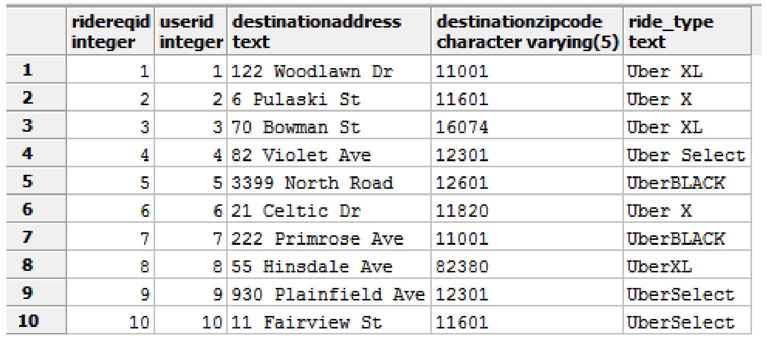
	payid integer	userid integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

Ride_Request Table

Displays information of what ride type is requested, and destination information such as address and zip code

```
CREATE TABLE Ride_Request(
rideReqID int NOT NULL,
userID int NOT NULL,
destinationAddress text NOT NULL,
destinationZipCode varchar(5) NOT NULL,
ride_type text NOT NULL,
PRIMARY KEY (rideReqID)
)
;
```

Functional Dependencies: rideReqID → userID, destinationAddress, destinationZipCode, ride_type



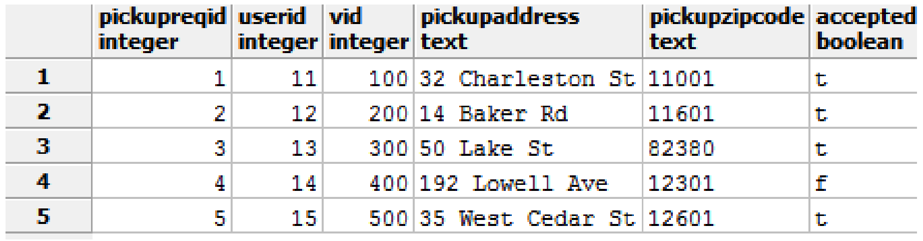
	ridereqid integer	userid integer	destinationaddress text	destinationzipcode character varying(5)	ride_type text
1	1	1	122 Woodlawn Dr	11001	Uber XL
2	2	2	6 Pulaski St	11601	Uber X
3	3	3	70 Bowman St	16074	Uber XL
4	4	4	82 Violet Ave	12301	Uber Select
5	5	5	3399 North Road	12601	UberBLACK
6	6	6	21 Celtic Dr	11820	Uber X
7	7	7	222 Primrose Ave	11001	UberBLACK
8	8	8	55 Hinsdale Ave	82380	UberXL
9	9	9	930 Plainfield Ave	12301	UberSelect
10	10	10	11 Fairview St	11601	UberSelect

Pickup_Request Table

Display pick up request information and such as the vehicle linked to the pickup, the pickup address and if the payment is accepted

```
CREATE TABLE Pickup_Request(
pickupReqID int NOT NULL,
userID int NOT NULL references Driver(userID),
VID int NOT NULL,
pickupAddress text NOT NULL,
pickupZipCode text NOT NULL,
accepted Boolean DEFAULT TRUE,
PRIMARY KEY (pickupReqID)
)
;
```

Functional Dependencies:
pickupReqID → userID, VID, pickupAddress, pickupZipCode, accepted



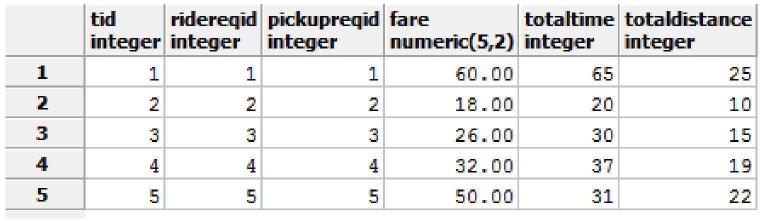
	pickupreqid integer	userid integer	vid integer	pickupaddress text	pickupzipcode text	accepted boolean
1	1	11	100	32 Charleston St	11001	t
2	2	12	200	14 Baker Rd	11601	t
3	3	13	300	50 Lake St	82380	t
4	4	14	400	192 Lowell Ave	12301	f
5	5	15	500	35 West Cedar St	12601	t

Trip Table

Displays fare price and distance of trip between locations

```
CREATE TABLE Trip(
TID int NOT NULL,
rideReqID int NOT NULL references Ride_Request(rideReqID),
pickupReqID int NOT NULL references Pickup_Request(pickupReqID),
fare decimal (5,2),
totalTime int NOT NULL,
totalDistance int NOT NULL,
PRIMARY KEY (TID)
)
;
```

Functional Dependencies:
TID → rideReqID, pickupReqID, fare, totalTime, totalDistance,



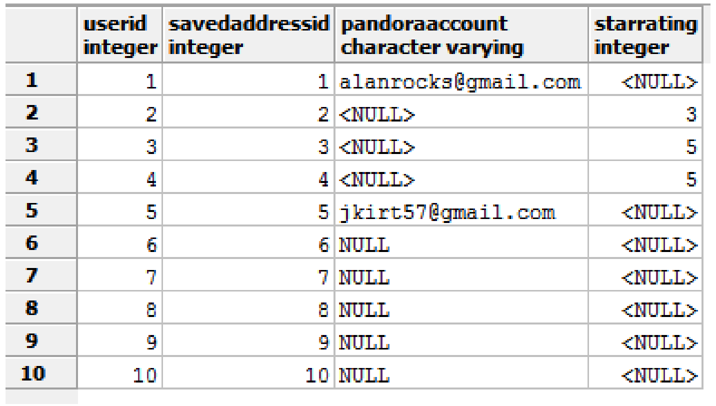
	tid integer	ridereqid integer	pickupreqid integer	fare numeric(5,2)	totaltime integer	totaldistance integer
1	1	1	1	60.00	65	25
2	2	2	2	18.00	20	10
3	3	3	3	26.00	30	15
4	4	4	4	32.00	37	19
5	5	5	5	50.00	31	22

Customer Table

Keeps track of individual users userID's, their saved addresses as savedAddressID, pandora account information (if available), and star rating

```
CREATE TABLE Customer(
userID int NOT NULL references User_Info(userID),
savedAddressID int NOT NULL,
pandoraAccount varchar,
starRating int,
PRIMARY KEY (userID)
)
;
```

Functional Dependencies:
userID → savedAddressID, pandoraAccount, starRating



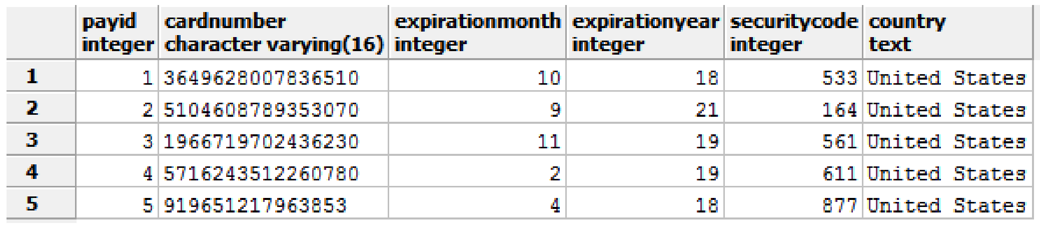
	userid integer	savedaddressid integer	pandoraaccount character varying	starrating integer
1	1	1	alanrocks@gmail.com	<NULL>
2	2	2	<NULL>	3
3	3	3	<NULL>	5
4	4	4	<NULL>	5
5	5	5	jkirt57@gmail.com	<NULL>
6	6	6	NULL	<NULL>
7	7	7	NULL	<NULL>
8	8	8	NULL	<NULL>
9	9	9	NULL	<NULL>
10	10	10	NULL	<NULL>

Card_Info Table

Keeps track of users credit card information

```
CREATE TABLE Card_Info(
payID int NOT NULL references Payment_Methods(payID),
cardNumber varchar(16) NOT NULL,
expirationMonth int NOT NULL,
expirationYear int NOT NULL,
securityCode int NOT NULL,
country text NOT NULL,
PRIMARY KEY (payID)
)
;
```

Functional Dependencies:
payID $\rightarrow$ cardNumber, expirationMonth, expirationYear, securityCode, country



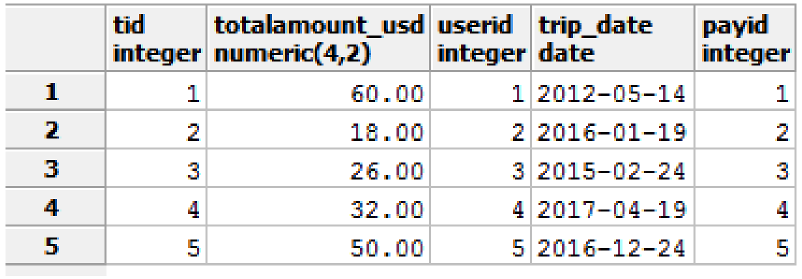
	payid integer	cardnumber character varying(16)	expirationmonth integer	expirationyear integer	securitycode integer	country text
1	1	3649628007836510	10	18	533	United States
2	2	5104608789353070	9	21	164	United States
3	3	1966719702436230	11	19	561	United States
4	4	5716243512260780	2	19	611	United States
5	5	919651217963853	4	18	877	United States

Customer_Trip_Invoice Table

Displays the price of each trip associated with each user by TID, userID, payID

```
CREATE TABLE Customer_Trip_Invoice(
TID int NOT NULL,
totalAmount_USD decimal (4,2) NOT NULL,
userID int NOT NULL,
trip_date date NOT NULL,
payID int NOT NULL,
PRIMARY KEY (TID)
)
;
```

Functional Dependencies: TID → totalAmount_USD, userID, trip_date, payID



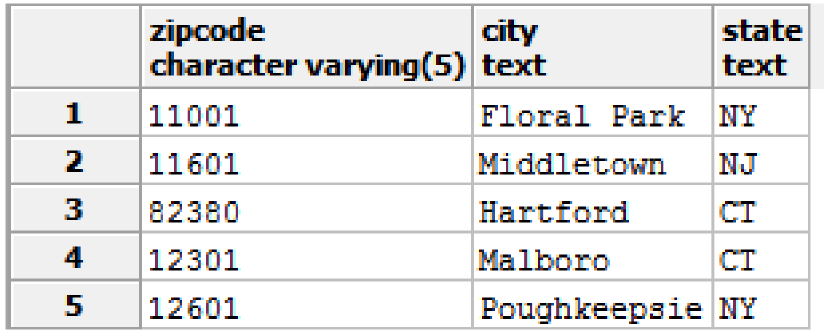
	tid integer	totalamount_usd numeric(4,2)	userid integer	trip_date date	payid integer
1	1	60.00	1	2012-05-14	1
2	2	18.00	2	2016-01-19	2
3	3	26.00	3	2015-02-24	3
4	4	32.00	4	2017-04-19	4
5	5	50.00	5	2016-12-24	5

Zip_Codes Table

Displays zip code information from addresses

Functional Dependencies: zipCode → city, state

```
CREATE TABLE Zip_Codes(
zipCode varchar(5) NOT NULL,
city text NOT NULL,
state text NOT NULL,
PRIMARY KEY (zipCode)
)
;
```



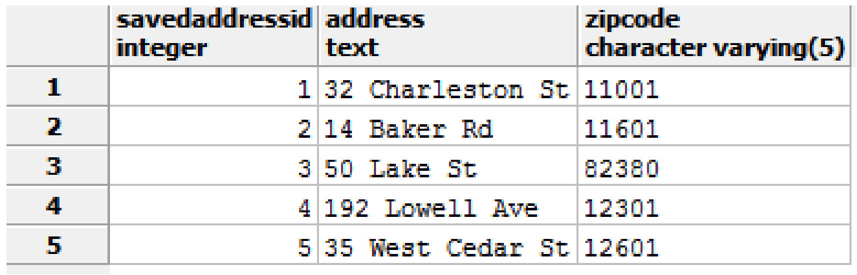
	zipcode character varying(5)	city text	state text
1	11001	Floral Park	NY
2	11601	Middletown	NJ
3	82380	Hartford	CT
4	12301	Malboro	CT
5	12601	Poughkeepsie	NY

Address_Book Table

Displays the saved location of the users original address entered when first signing up for an account

```
CREATE TABLE Address_Book(
savedAddressID int NOT NULL,
address text NOT NULL,
zipCode varchar(5) references Zip_Codes(zipCode) NOT NULL,
PRIMARY KEY (savedAddressID)
)
;
```

Functional Dependencies:
savedAddressID → address, zipCode



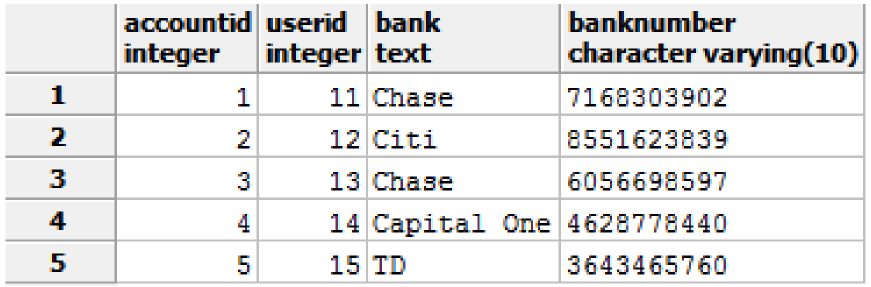
	savedaddressid integer	address text	zipcode character varying(5)
1	1	32 Charleston St	11001
2	2	14 Baker Rd	11601
3	3	50 Lake St	82380
4	4	192 Lowell Ave	12301
5	5	35 West Cedar St	12601

Driver_Accounts Table

Displays bank information of drivers

Functional Dependencies:
accountID→ userID, bank, bankNumber

```
CREATE TABLE Driver_Accounts(
accountID int NOT NULL,
userID int NOT NULL,
bank text NOT NULL,
bankNumber varchar(10),
PRIMARY KEY (accountID)
)
;
```



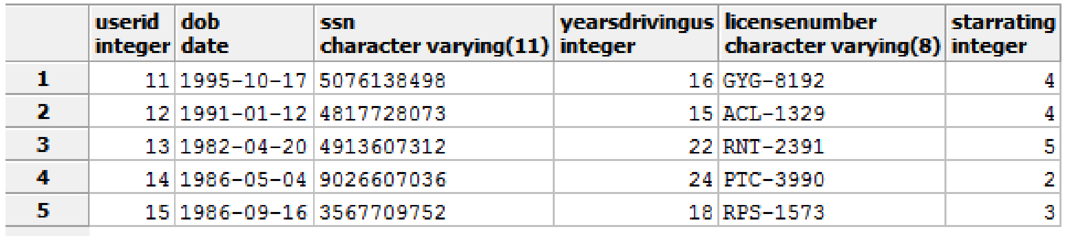
	accountid integer	userid integer	bank text	banknumber character varying(10)
1	1	11	Chase	7168303902
2	2	12	Citi	8551623839
3	3	13	Chase	6056698597
4	4	14	Capital One	4628778440
5	5	15	TD	3643465760

Driver Table

Displays personal information for all drivers

```
CREATE TABLE Driver(
userID int NOT NULL,
DOB date NOT NULL,
SSN varchar(11),
yearsDrivingUS int NOT NULL,
licenseNumber varchar(8) NOT NULL,
starRating int,
PRIMARY KEY (userID)
)
;
```

Functional Dependencies:
userID $\rightarrow$ DOB, SSN, yearsDrivingUS, licenseNumber, starRating



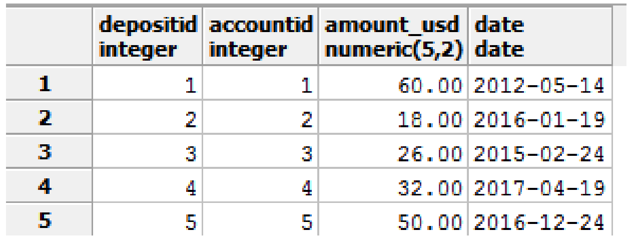
	userid integer	dob date	ssn character varying(11)	yearsdrivingus integer	licensenumber character varying(8)	starrating integer
1	11	1995-10-17	5076138498	16	GYG-8192	4
2	12	1991-01-12	4817728073	15	ACL-1329	4
3	13	1982-04-20	4913607312	22	RNT-2391	5
4	14	1986-05-04	9026607036	24	PTC-3990	2
5	15	1986-09-16	3567709752	18	RPS-1573	3

Driver_Deposits Table

Displays the amount drivers make and the date it is made on

Functional Dependencies: depositID → accountID, amount_USD, date

```
CREATE TABLE Driver_Deposits(
depositID int NOT NULL,
accountID int NOT NULL,
amount_USD decimal (5,2) NOT NULL,
date date NOT NULL,
PRIMARY KEY (depositID)
)
;
```



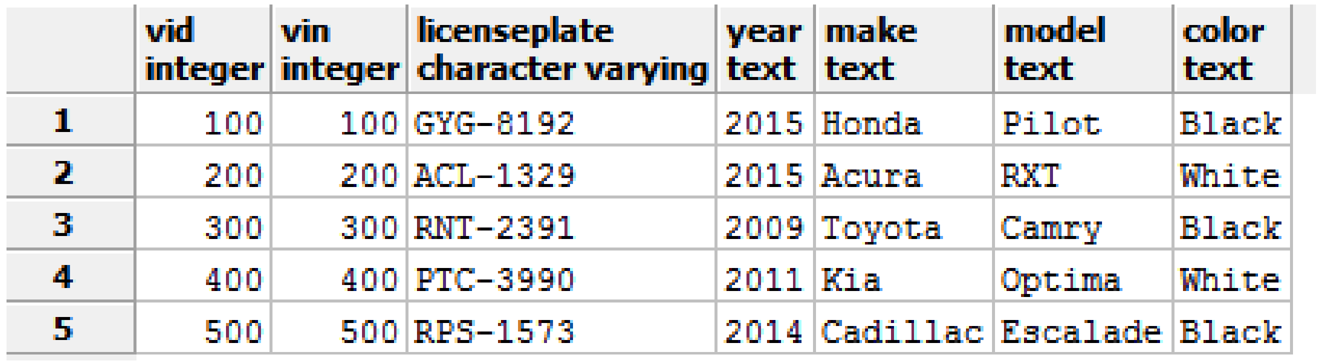
	depositid integer	accountid integer	amount_usd numeric(5,2)	date date
1	1	1	60.00	2012-05-14
2	2	2	18.00	2016-01-19
3	3	3	26.00	2015-02-24
4	4	4	32.00	2017-04-19
5	5	5	50.00	2016-12-24

Vehicle Table

Displays information relating to drivers vehicles

```
CREATE TABLE Vehicle(
VID int NOT NULL,
VIN int NOT NULL,
licensePlate varchar NOT NULL,
year text NOT NULL,
make text NOT NULL,
model text NOT NULL,
color text NOT NULL,
PRIMARY KEY (VID)
)
;
```

Functional Dependencies:
VID $\rightarrow$ VIN, licensePlate, year, make, model, color



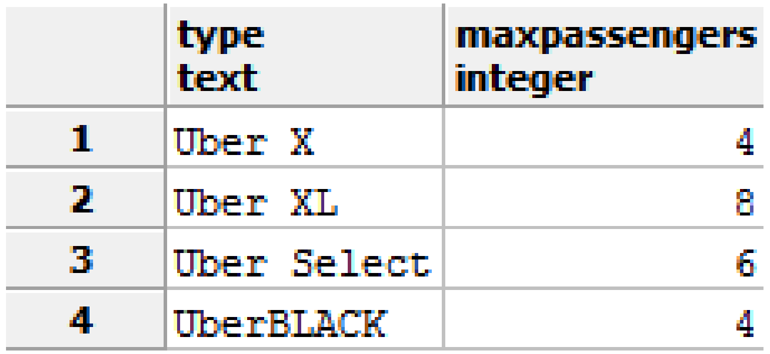
	vid integer	vin integer	licenseplate character varying	year text	make text	model text	color text
1	100	100	GYG-8192	2015	Honda	Pilot	Black
2	200	200	ACL-1329	2015	Acura	RXT	White
3	300	300	RNT-2391	2009	Toyota	Camry	Black
4	400	400	PTC-3990	2011	Kia	Optima	White
5	500	500	RPS-1573	2014	Cadillac	Escalade	Black

Classifications Table

Displays maximum amount of passengers for each Uber type

Functional Dependencies:
type $\rightarrow$maxPassengers

```
CREATE TABLE Classifications(
type text NOT NULL,
maxPassengers int NOT NULL,
PRIMARY KEY (type)
)
;
```



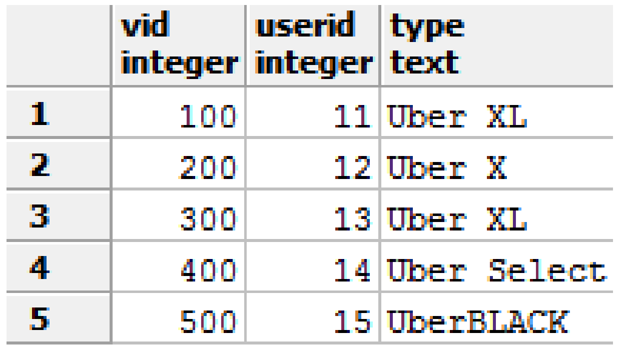
	type text	maxpassengers integer
1	Uber X	4
2	Uber XL	8
3	Uber Select	6
4	UberBLACK	4

Vehicle_Records Table

Displays Uber car type according to the VID and the appropriate userID

Functional Dependencies:
VID → userID, type

```
CREATE TABLE Vehicle_Records(
VID int NOT NULL,
userID int NOT NULL references Driver(userID),
type text NOT NULL references Classifications(type),
PRIMARY KEY (VID)
)
;
```

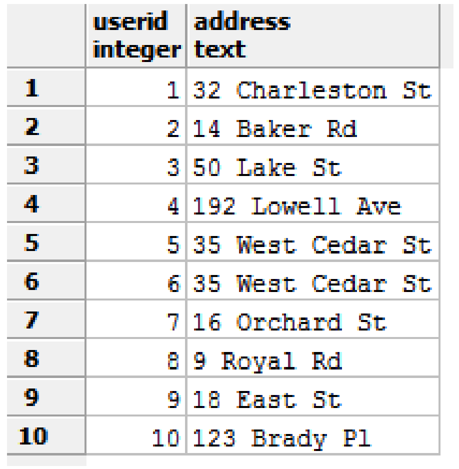


	vid integer	userid integer	type text
1	100	11	Uber XL
2	200	12	Uber X
3	300	13	Uber XL
4	400	14	Uber Select
5	500	15	UberBLACK

Views

Creates view that simply displays users userID from customers table to their saved address in the address_book table

```
CREATE VIEW showAddress
AS SELECT Customer.userID, Address_Book.address
FROM Customer, Address_Book
WHERE customer.savedAddressID = address_book.savedAddressID;
```

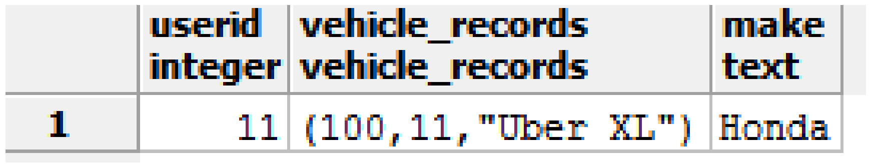


	userid integer	address text
1	1	32 Charleston St
2	2	14 Baker Rd
3	3	50 Lake St
4	4	192 Lowell Ave
5	5	35 West Cedar St
6	6	35 West Cedar St
7	7	16 Orchard St
8	8	9 Royal Rd
9	9	18 East St
10	10	123 Brady Pl

Views

Creates View of information of the driver with a userID that correlates to a to the vehicle make of a *Honda* and its vehicle type

```
CREATE VIEW DriverInfo
AS SELECT Driver.userID, Vehicle_records, Vehicle.make
FROM Driver,Vehicle_Records, Vehicle
WHERE Driver.userID = vehicle_records.userID
  AND vehicle_records.vid = vehicle.vid
  AND vehicle.make= 'Honda';
```

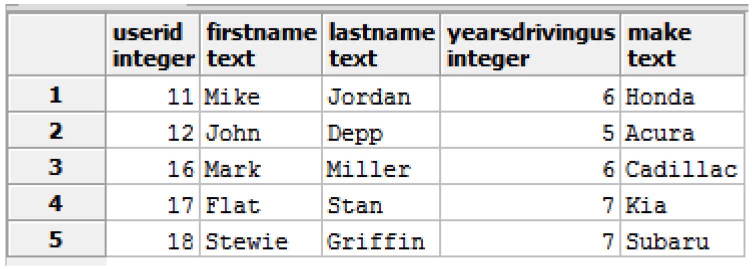


	userid integer	vehicle_records vehicle_records	make text
1	11	(100,11,"Uber XL")	Honda

Report

Joins tables user_info, driver, vehicle_records, and vehicle tables and displays userID, firstname, lastname yearsdrivingus and the make of the car each driver drives

```
SELECT u.userID, u.firstName, u.lastName, d.yearsDrivingUS, v.make
FROM user_info u inner join driver d on u.userID = d.userID
                inner join vehicle_records vr on d.userID = vr.userID
                inner join vehicle v on vr.vid = v.vid
WHERE d.yearsdrivingus > 4;
```

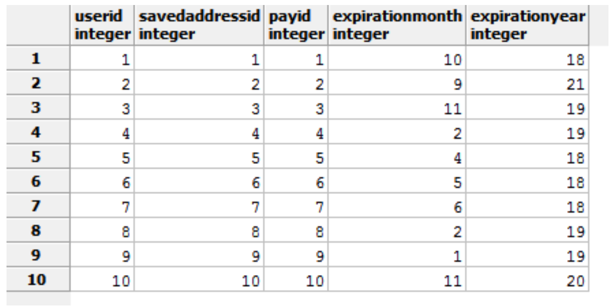


	userid integer	firstname text	lastname text	yearsdrivingus integer	make text
1	11	Mike	Jordan	6	Honda
2	12	John	Depp	5	Acura
3	16	Mark	Miller	6	Cadillac
4	17	Flat	Stan	7	Kia
5	18	Stewie	Griffin	7	Subaru

Report

Joins customer table on payment_methods table on card_info table to show userID, savedAddressID, payID, expirationMonth, expirationYear

```
SELECT c.userID , c.savedAddressID, p.payID, ci.expirationMonth, ci.expirationYear
FROM Customer c inner join payment_methods p on c.userID = p.userID
                    inner join Card_Info ci on p.payID = ci.payID
```



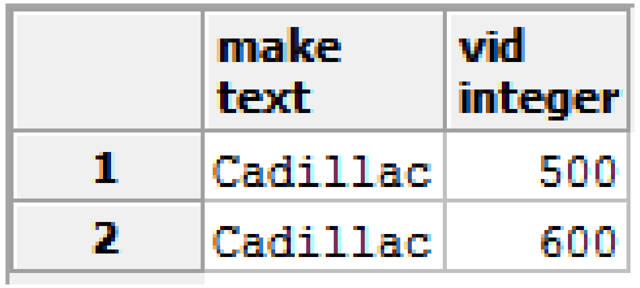
	userid integer	savedaddressid integer	payid integer	expirationmonth integer	expirationyear integer
1	1	1	1	10	18
2	2	2	2	9	21
3	3	3	3	11	19
4	4	4	4	2	19
5	5	5	5	4	18
6	6	6	6	5	18
7	7	7	7	6	18
8	8	8	8	2	19
9	9	9	9	1	19
10	10	10	10	11	20

Stored Procedure

Creates trigger to direct you to vehicles with VID’s according to the make being Cadillac

```
CREATE OR REPLACE FUNCTION carTypeFor(text, refcursor) RETURNS refcursor AS
$$
DECLARE
  type text             :=$1;
  resultset refcursor :=$2;
BEGIN OPEN resultset for
                    select make, vid
                    from Vehicle
                    where vehicle.make = 'Cadillac';
                    return resultset;
end;
$$
language plpgsql;

select carTypeFor ('Cadillac', 'results');
Fetch all from results;
```



	make text	vid integer
1	Cadillac	500
2	Cadillac	600

Security

For security reasons, the best way to implement measures on a system this large is to use hashed passwords, making it difficult for intruders. Uber’s network administrators would also grant access for drivers to accept or deny rides if they are in the area. Passengers are limited to changing basic profile information, payment information and can book rides, but do not have access to personal information of drivers. Network administrators are granted access to the whole system.

Security

```
Create role admin;
Create role driver;
Create role passenger;

-- Admin
Grant all on all tables in schema public to admin

-- Passenger
Grant insert, update on Card_Info to passenger
Grant insert, update on User_Info to passenger

--Driver
Grant insert, update on Driver_Accounts to driver
Grant insert, update on vehicle to driver
Grant insert, update on Pickup_Request to driver
Grant insert, update on Ride_Request to driver
```

Implementation Notes

- If users were given the opportunity to pay with different payment methods, more tables should be created to differentiate account types
- This separation of payment information would reduce the amount of users entered in each table or encourage more users to join as there is a greater amount of payment options

Known Problems

- Drivers may own more than one car thus adding more VID's and vehicle information which adds redundancy in linking VID's to userID's
- Passengers are unaware of how many years the driver has been driving in the U.S.

Future Enhancements

- In depth profile view of driver
- Passengers should be able to pay cash upon arrival of driver instead of set up a credit account
- Show what drivers in the area are already driving passengers