Car Wash

Database Design Proposal

By: Katerina Tzannes

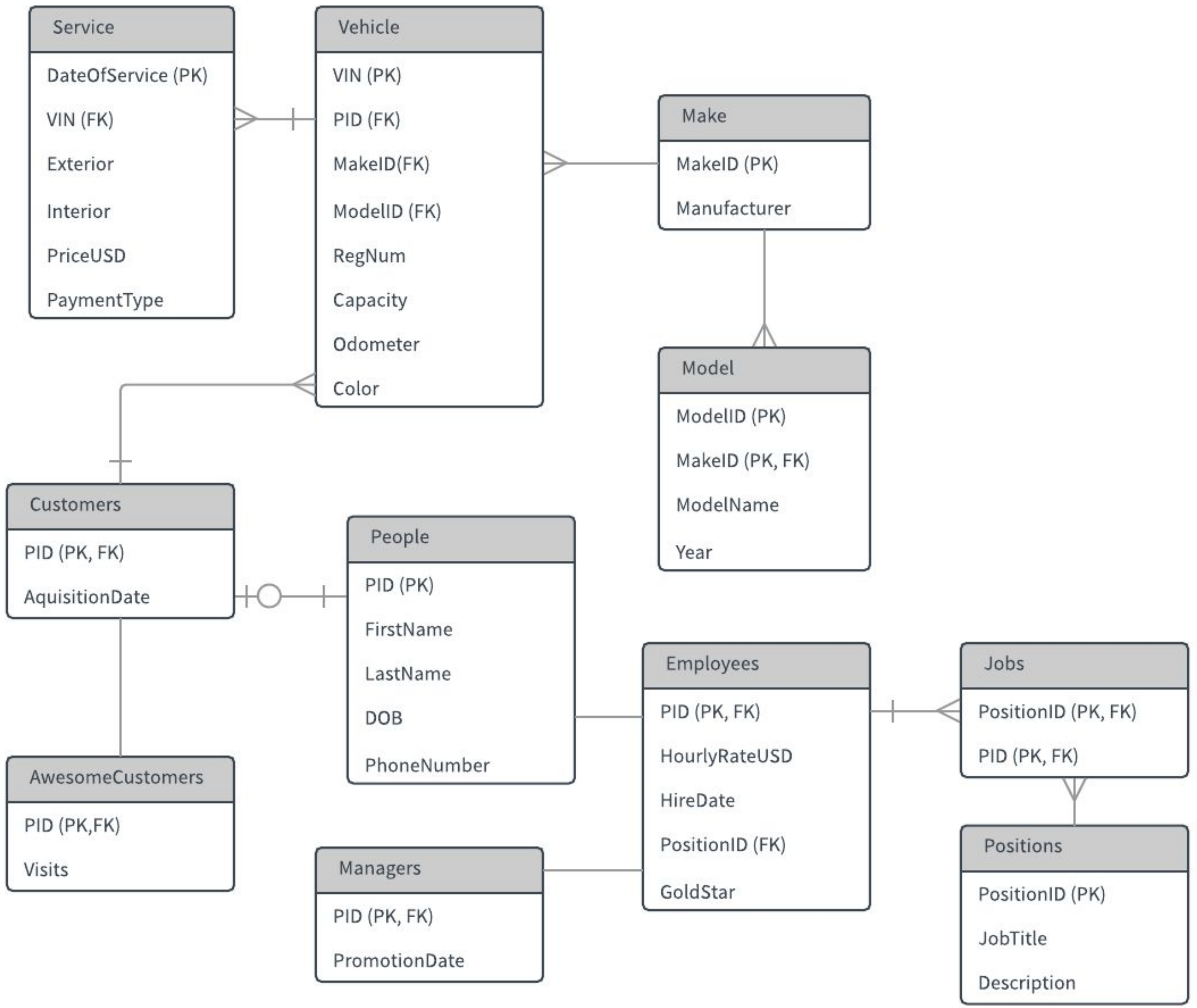
Table of Contents

Executive Summary

The car wash database keeps track of people, such as employees and customers, with of course all of their precious data, and obviously their vehicles. This paper explains all the tables in the database and why they are important. The database is in third normal form, each table is unique and has specific fields in order to ensure the businesses operation will always run smoothly.

The order of this paper starts off with the ER Diagram, followed by the tables, queries/reports, views, stored procedures, and triggers. Nothing is perfect and in that case there is always room for improvement.

ER- Diagram



Service
DateOfService (PK)
VIN (FK)
Exterior
Interior
PriceUSD
PaymentType
Vehicle
VIN (PK)
PID (FK)
MakeID(FK)
ModelID (FK)
RegNum
Capacity
Odometer
Color
Make
MakeID (PK)
Manufacturer
Model
ModelID (PK)
MakeID (PK, FK)
ModelName
Year
Customers
PID (PK, FK)
AquisitionDate
People
PID (PK)
FirstName
LastName
DOB
PhoneNumber
Employees
PID (PK, FK)
HourlyRateUSD
HireDate
PositionID (FK)
GoldStar
Jobs
PositionID (PK, FK)
PID (PK, FK)
Positions
PositionID (PK)
JobTitle
Description
AwesomeCustomers
PID (PK,FK)
Visits
Managers
PID (PK, FK)
PromotionDate

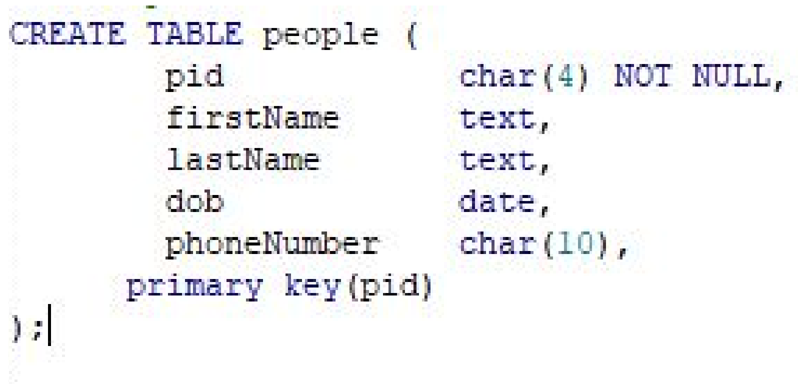
People Table

Consists of all people and their information, like their name, date of birth, and phone number. It is based on their unique pid.

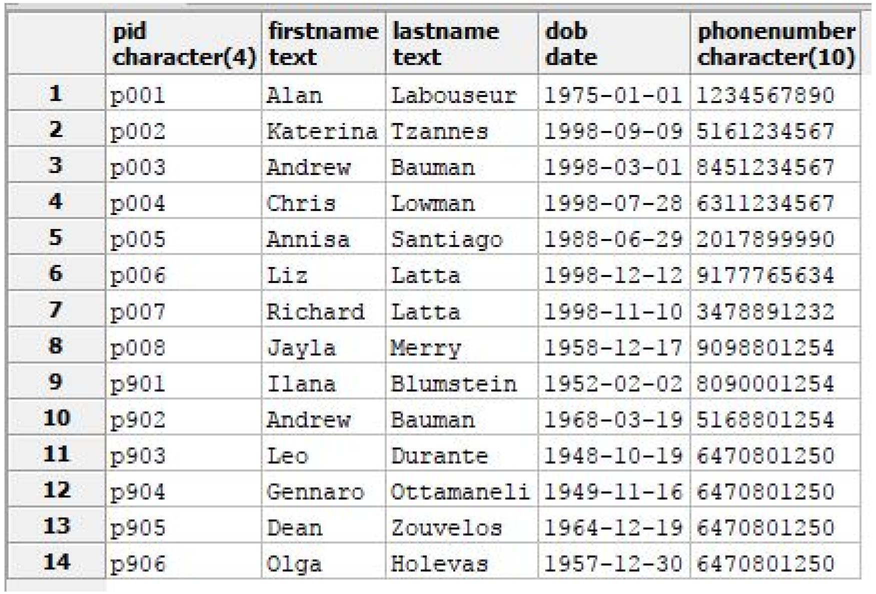
Functional Dependency:

pid -> firstName, lastName, DOB,

phoneNumber



```
CREATE TABLE people (
        pid             char(4) NOT NULL,
        firstName       text,
        lastName        text,
        dob             date,
        phoneNumber     char(10),
      primary key(pid)
);
```



	pid character(4)	firstname text	lastname text	dob date	phonenumber character(10)
1	p001	Alan	Labouseur	1975-01-01	1234567890
2	p002	Katerina	Tzannes	1998-09-09	5161234567
3	p003	Andrew	Bauman	1998-03-01	8451234567
4	p004	Chris	Lowman	1998-07-28	6311234567
5	p005	Annisa	Santiago	1988-06-29	2017899990
6	p006	Liz	Latta	1998-12-12	9177765634
7	p007	Richard	Latta	1998-11-10	3478891232
8	p008	Jayla	Merry	1958-12-17	9098801254
9	p901	Ilana	Blumstein	1952-02-02	8090001254
10	p902	Andrew	Bauman	1968-03-19	5168801254
11	p903	Leo	Durante	1948-10-19	6470801250
12	p904	Gennaro	Ottamaneli	1949-11-16	6470801250
13	p905	Dean	Zouvelos	1964-12-19	6470801250
14	p906	Olga	Holevas	1957-12-30	6470801250

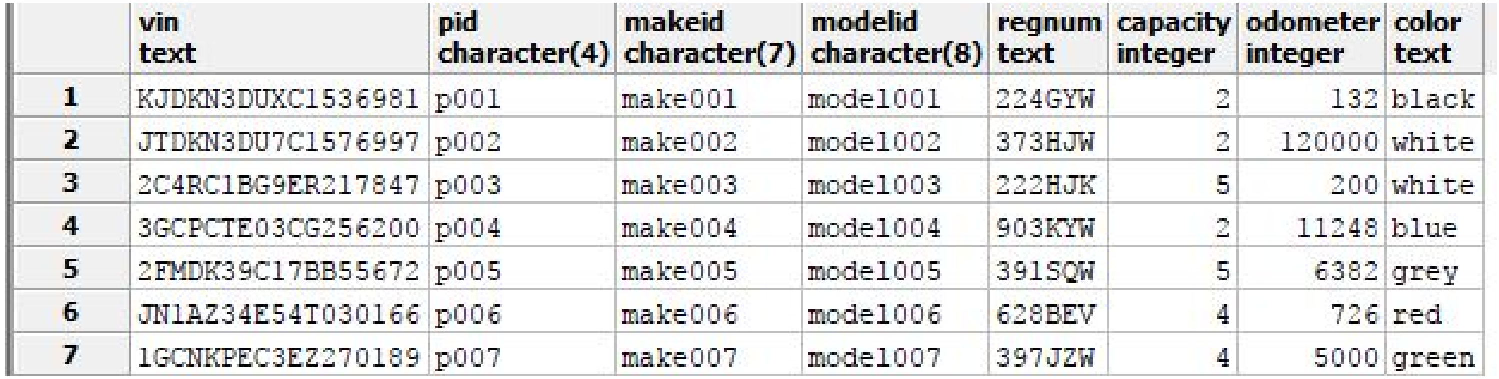
Vehicle Table

Contains an individual's vehicle information, weather they are an employee or customer.

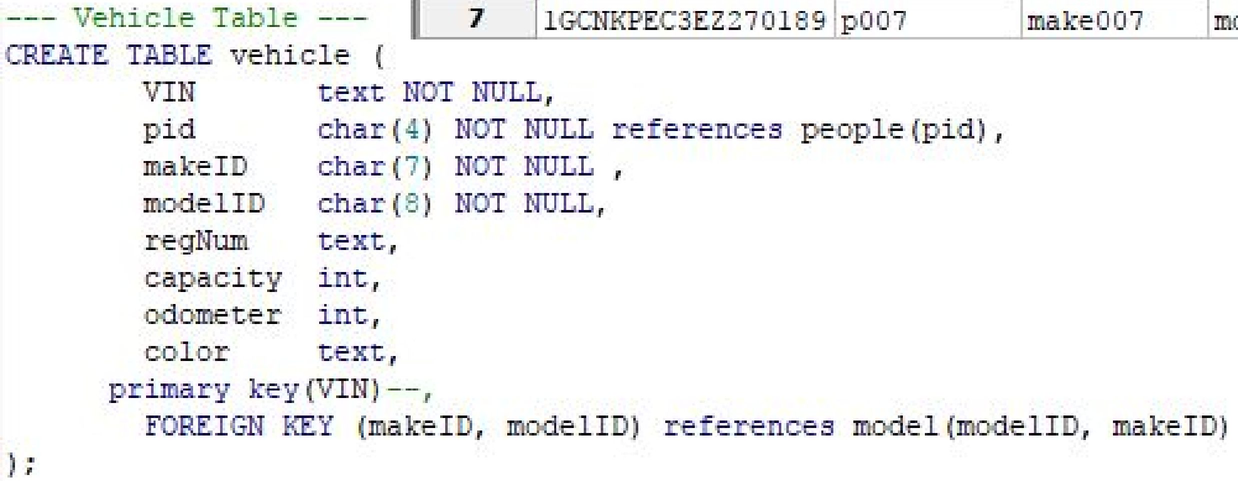
Functional Dependencies:

VIN -> PID, MakeID, ModelID, RegNum, Capacity, Odometer, Color

Vehicle Table cont.



	vin text	pid character(4)	makeid character(7)	modelid character(8)	regnum text	capacity integer	odometer integer	color text
1	KJDKN3DUXC1536981	p001	make001	model001	224GYW	2	132	black
2	JTDKN3DU7C1576997	p002	make002	model002	373HJW	2	120000	white
3	2C4RC1BG9ER217847	p003	make003	model003	222HJK	5	200	white
4	3GCPCTE03CG256200	p004	make004	model004	903KYW	2	11248	blue
5	2FMDK39C17BB55672	p005	make005	model005	391SQW	5	6382	grey
6	JN1AZ34E54T030166	p006	make006	model006	628BEV	4	726	red
7	1GCNKPEC3EZ270189	p007	make007	model007	397JZW	4	5000	green



```
--- Vehicle Table ---
CREATE TABLE vehicle (
        VIN        text NOT NULL,
        pid        char(4) NOT NULL references people(pid),
        makeID     char(7) NOT NULL ,
        modelID    char(8) NOT NULL,
        regNum     text,
        capacity   int,
        odometer   int,
        color      text,
      primary key(VIN)--,
        FOREIGN KEY (makeID, modelID) references model(modelID, makeID)
);
```

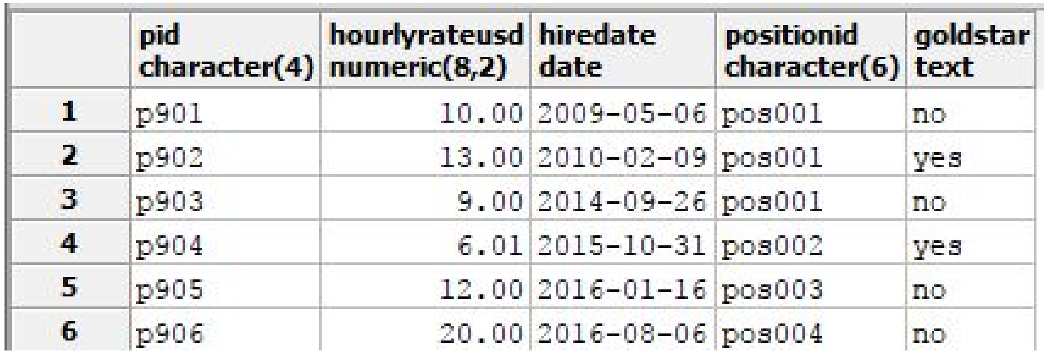
Employees Table

Employees are all people, thus must be in the people table. Consists of all employees and their hourlyRateUSD, hireDate, positionID, and if they have received a goldStar. If there is no information about whether an employee has ever received a goldStar the default value is no.

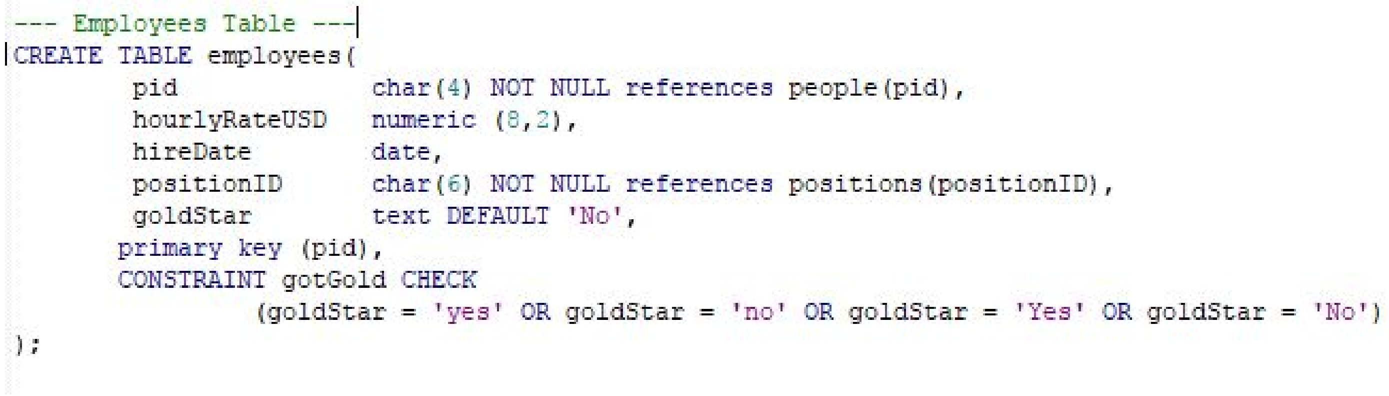
Functional Dependency:

PID -> hourlyRateUSD, hireDate, positionID, goldStar

Employees Table Cont.



	pid character(4)	hourlyrateusd numeric(8,2)	hiredate date	positionid character(6)	goldstar text
1	p901	10.00	2009-05-06	pos001	no
2	p902	13.00	2010-02-09	pos001	yes
3	p903	9.00	2014-09-26	pos001	no
4	p904	6.01	2015-10-31	pos002	yes
5	p905	12.00	2016-01-16	pos003	no
6	p906	20.00	2016-08-06	pos004	no



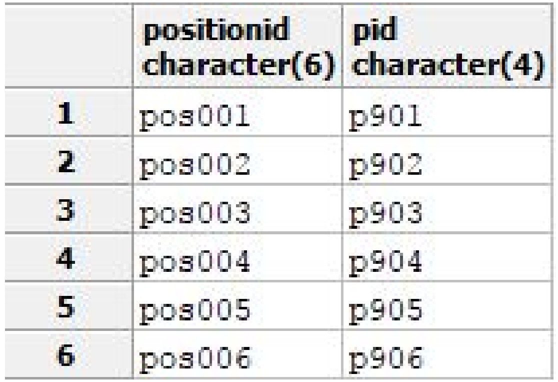
```
--- Employees Table ---
CREATE TABLE employees(
        pid             char(4) NOT NULL references people(pid),
        hourlyRateUSD   numeric (8,2),
        hireDate        date,
        positionID      char(6) NOT NULL references positions(positionID),
        goldStar        text DEFAULT 'No',
       primary key (pid),
       CONSTRAINT gotGold CHECK
                (goldStar = 'yes' OR goldStar = 'no' OR goldStar = 'Yes' OR goldStar = 'No')
);
```

Jobs Table

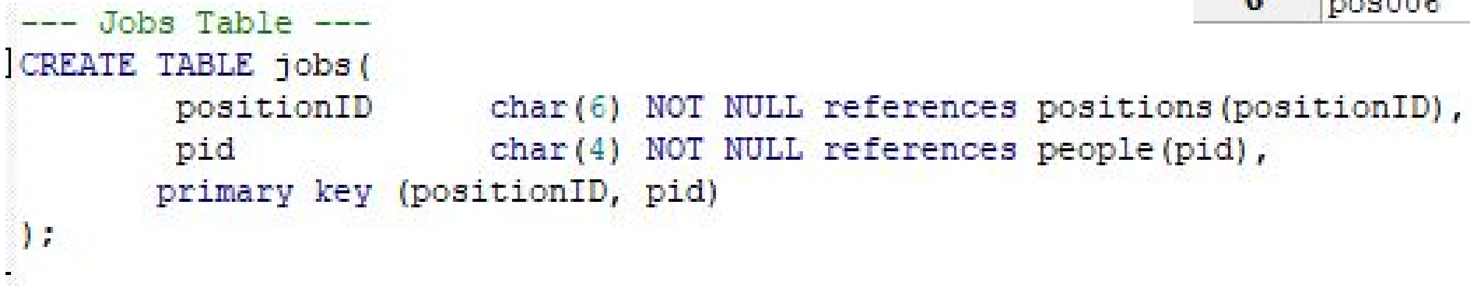
The job table contains the pid and positionID of an employee. One employee can have multiple positions and multiple positions can be dispersed amongst many employees.

Functional Dependency:

pid, positionID ->



	positionid character(6)	pid character(4)
1	pos001	p901
2	pos002	p902
3	pos003	p903
4	pos004	p904
5	pos005	p905
6	pos006	p906



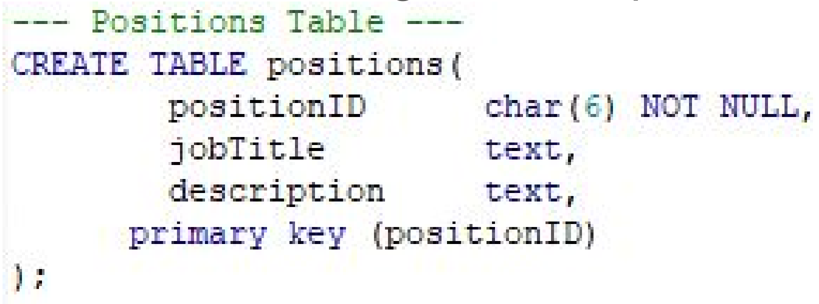
```
--- Jobs Table ---
CREATE TABLE jobs(
        positionID      char(6) NOT NULL references positions(positionID),
        pid             char(4) NOT NULL references people(pid),
       primary key (positionID, pid)
);
```

Positions Table

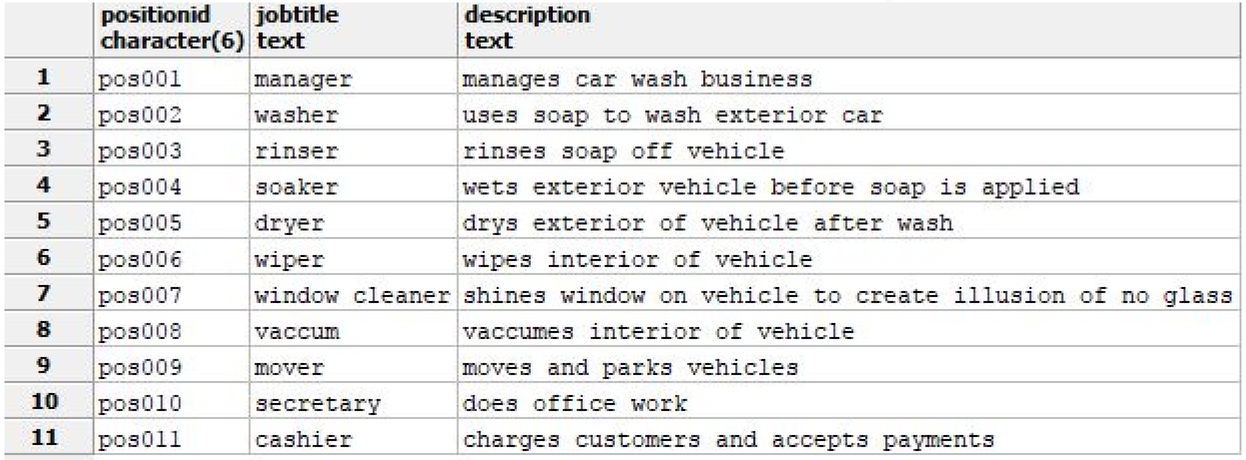
Holds all the existing job titles with a description, which are assigned a unique positionID.

Functional Dependency:

positionID -> jobTitle, description



```
--- Positions Table ---
CREATE TABLE positions(
        positionID      char(6) NOT NULL,
        jobTitle        text,
        description     text,
      primary key (positionID)
);
```



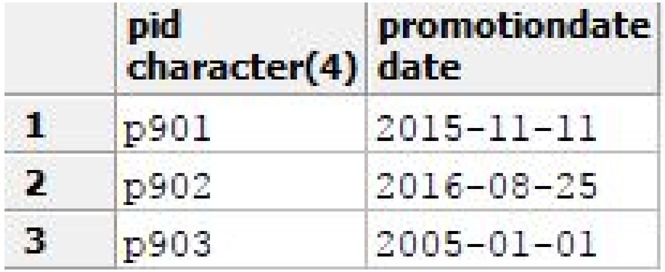
	positionid character(6)	jobtitle text	description text
1	pos001	manager	manages car wash business
2	pos002	washer	uses soap to wash exterior car
3	pos003	rinser	rinses soap off vehicle
4	pos004	soaker	wets exterior vehicle before soap is applied
5	pos005	dryer	drys exterior of vehicle after wash
6	pos006	wiper	wipes interior of vehicle
7	pos007	window cleaner	shines window on vehicle to create illusion of no glass
8	pos008	vaccum	vaccumes interior of vehicle
9	pos009	mover	moves and parks vehicles
10	pos010	secretary	does office work
11	pos011	cashier	charges customers and accepts payments

Managers Table

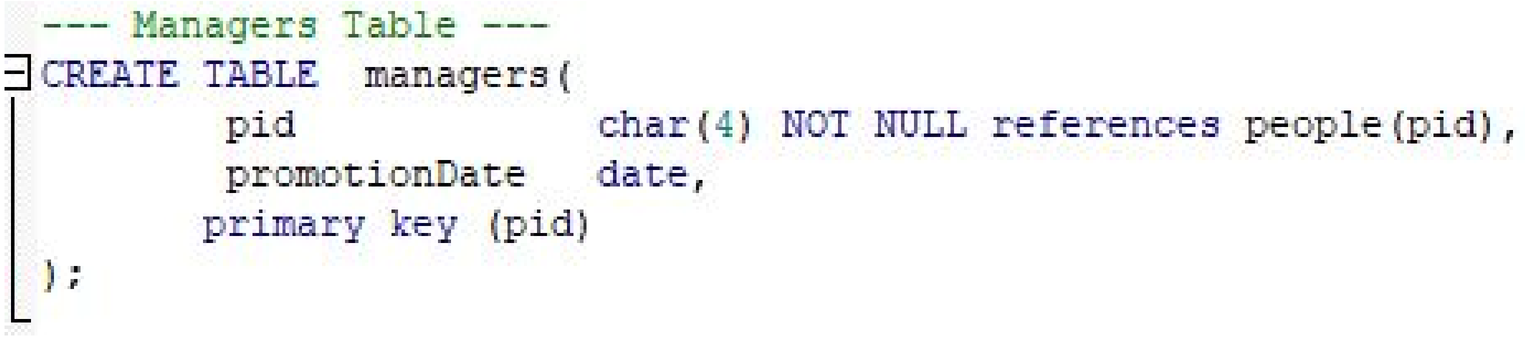
This table is specifically for managers and contains pid and promotionDate for each manager.

Functional Dependency:

PID -> promotionDate



	pid character(4)	promotiondate date
1	p901	2015-11-11
2	p902	2016-08-25
3	p903	2005-01-01



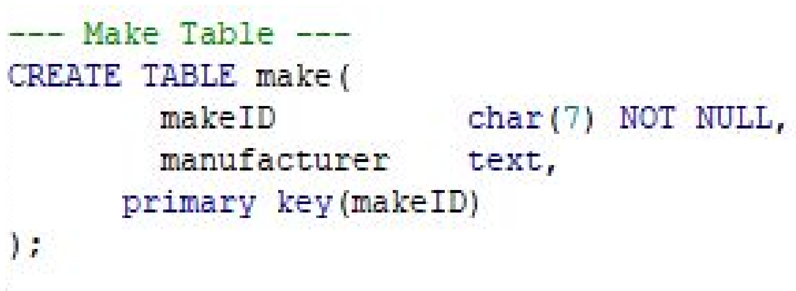
```
--- Managers Table ---
CREATE TABLE  managers(
        pid             char(4) NOT NULL references people(pid),
        promotionDate   date,
       primary key (pid)
);
```

Make Table

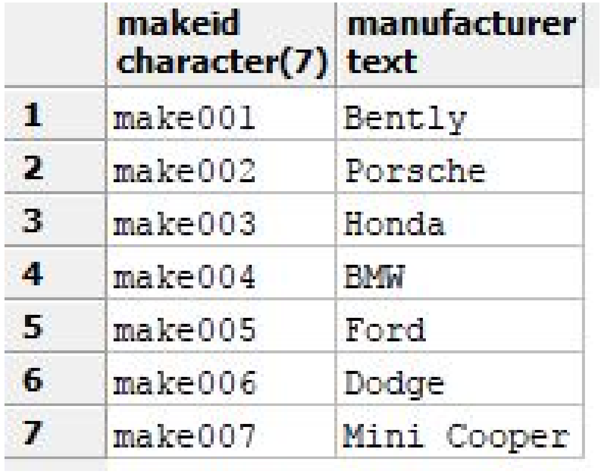
Contains all the manufacturers that make vehicles and their unique makeID. A make can have many models, however a vehicle can only have one make.

Functional Dependency:

makeID -> manufacturer



```
--- Make Table ---
CREATE TABLE make(
        makeID          char(7) NOT NULL,
        manufacturer    text,
      primary key(makeID)
);
```



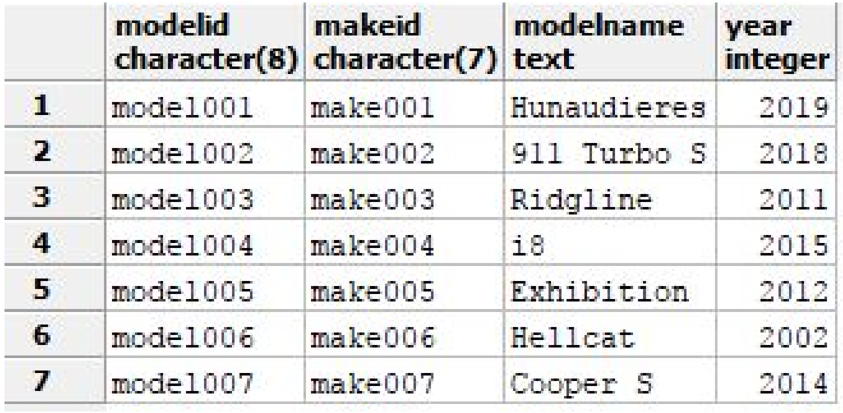
	makeid character(7)	manufacturer text
1	make001	Bently
2	make002	Porsche
3	make003	Honda
4	make004	BMW
5	make005	Ford
6	make006	Dodge
7	make007	Mini Cooper

Model Table

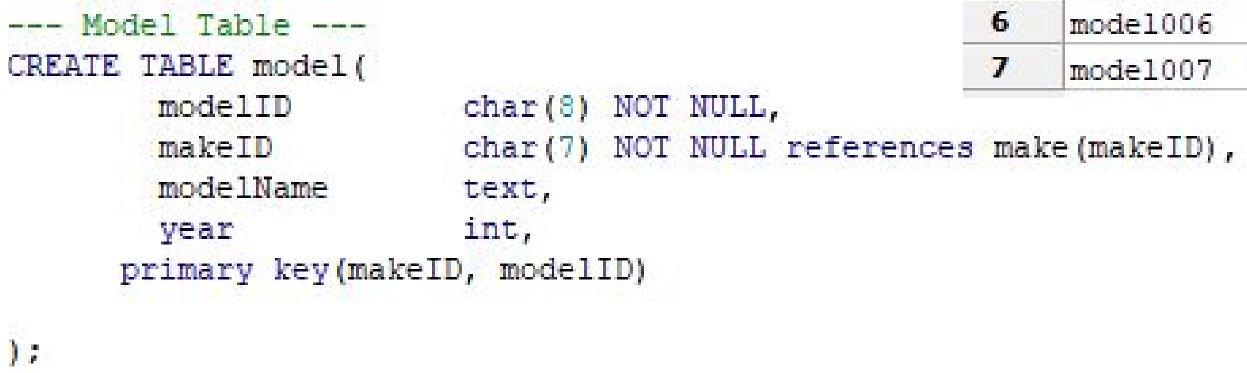
A vehicle make can have many models. The table consists of a unique modelID, makeId, modelName, and year.

Functional Dependency:

modelID, makeID -> modelName, year



	modelid character(8)	makeid character(7)	modelname text	year integer
1	model001	make001	Hunaudieres	2019
2	model002	make002	911 Turbo S	2018
3	model003	make003	Ridgline	2011
4	model004	make004	i8	2015
5	model005	make005	Exhibition	2012
6	model006	make006	Hellcat	2002
7	model007	make007	Cooper S	2014



```
--- Model Table ---
CREATE TABLE model(
        modelID         char(8) NOT NULL,
        makeID          char(7) NOT NULL references make(makeID),
        modelName       text,
        year            int,
      primary key(makeID, modelID)

);
```

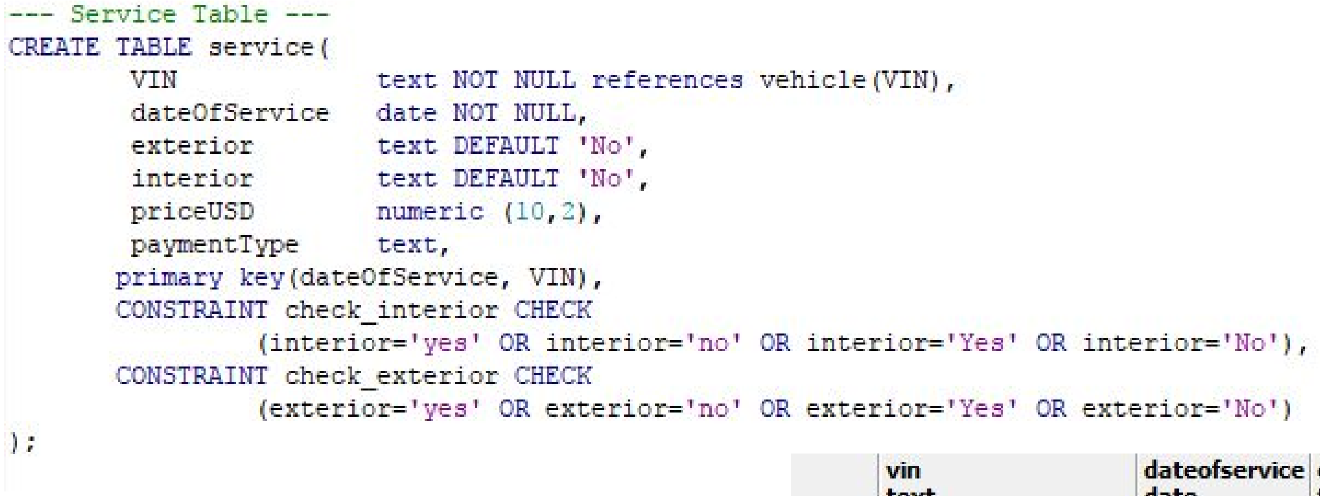
Service Table

The service table consists of VIN and dateOfService as primary keys, exterior, interior, priceUSD, and paymentType.

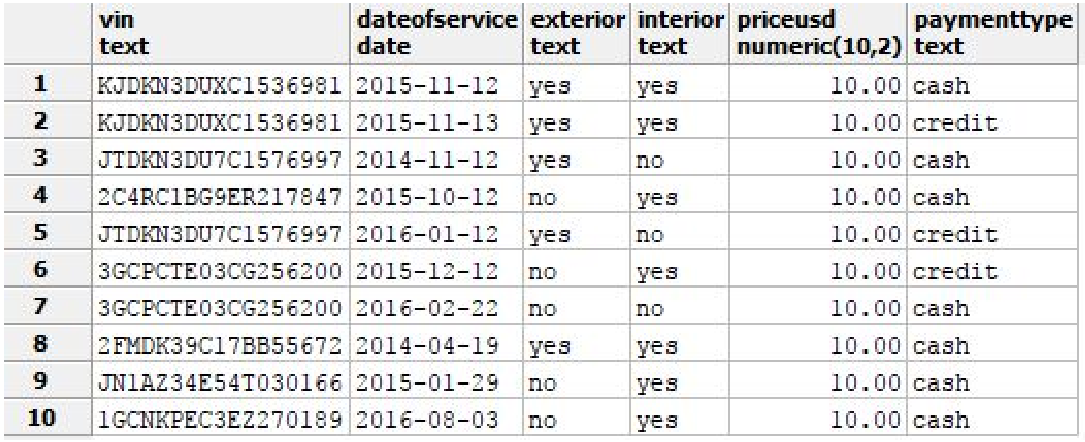
Functional Dependency:

VIN, dateOfService -> exterior, interior, priceUSD, paymentType

Service Table Cont.



```
--- Service Table ---
CREATE TABLE service(
        VIN              text NOT NULL references vehicle(VIN),
        dateOfService    date NOT NULL,
        exterior         text DEFAULT 'No',
        interior         text DEFAULT 'No',
        priceUSD         numeric (10,2),
        paymentType      text,
       primary key(dateOfService, VIN),
       CONSTRAINT check_interior CHECK
                (interior='yes' OR interior='no' OR interior='Yes' OR interior='No'),
       CONSTRAINT check_exterior CHECK
                (exterior='yes' OR exterior='no' OR exterior='Yes' OR exterior='No')
);
```



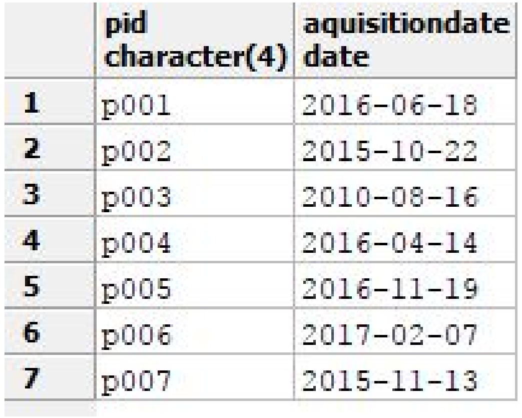
	vin text	dateofservice date	exterior text	interior text	priceusd numeric(10,2)	paymenttype text
1	KJDKN3DUXC1536981	2015-11-12	yes	yes	10.00	cash
2	KJDKN3DUXC1536981	2015-11-13	yes	yes	10.00	credit
3	JTDKN3DU7C1576997	2014-11-12	yes	no	10.00	cash
4	2C4RC1BG9ER217847	2015-10-12	no	yes	10.00	cash
5	JTDKN3DU7C1576997	2016-01-12	yes	no	10.00	credit
6	3GCPCTE03CG256200	2015-12-12	no	yes	10.00	credit
7	3GCPCTE03CG256200	2016-02-22	no	no	10.00	cash
8	2FMDK39C17BB55672	2014-04-19	yes	yes	10.00	cash
9	JN1AZ34E54T030166	2015-01-29	no	yes	10.00	cash
10	1GCNKPEC3EZ270189	2016-08-03	no	yes	10.00	cash

Customers Table

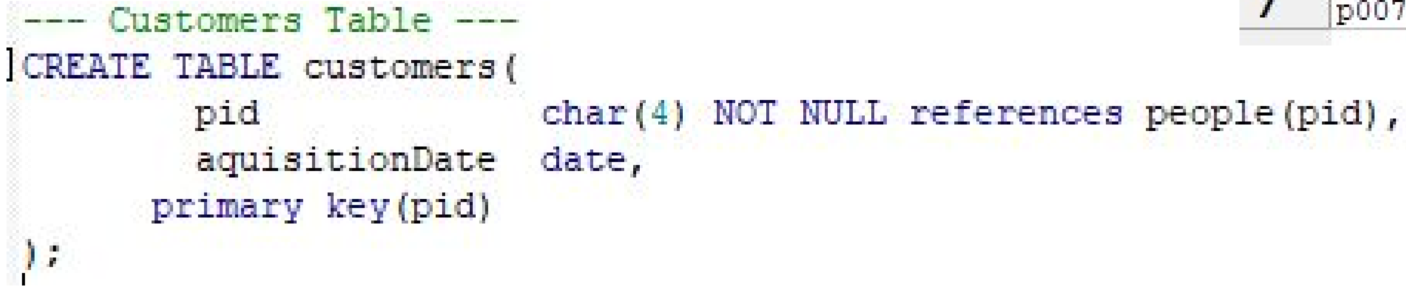
In order for a person to be a customer they must be in the people table and an employee may be a customer. One customer may have multiple vehicles. The table consists of PID and aquisitionDate

Functional Dependency:

Pid -> aquisitionDate



	pid character(4)	aquisitiondate date
1	p001	2016-06-18
2	p002	2015-10-22
3	p003	2010-08-16
4	p004	2016-04-14
5	p005	2016-11-19
6	p006	2017-02-07
7	p007	2015-11-13



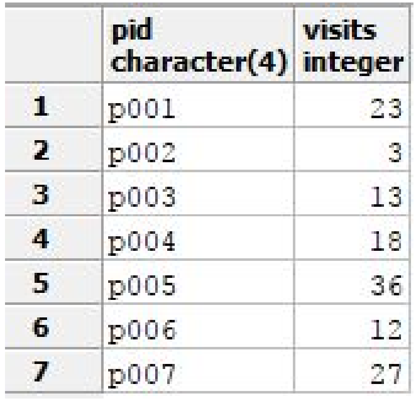
```
--- Customers Table ---
CREATE TABLE customers(
        pid             char(4) NOT NULL references people(pid),
        aquisitionDate  date,
      primary key(pid)
);
```

AwesomeCustomers Table

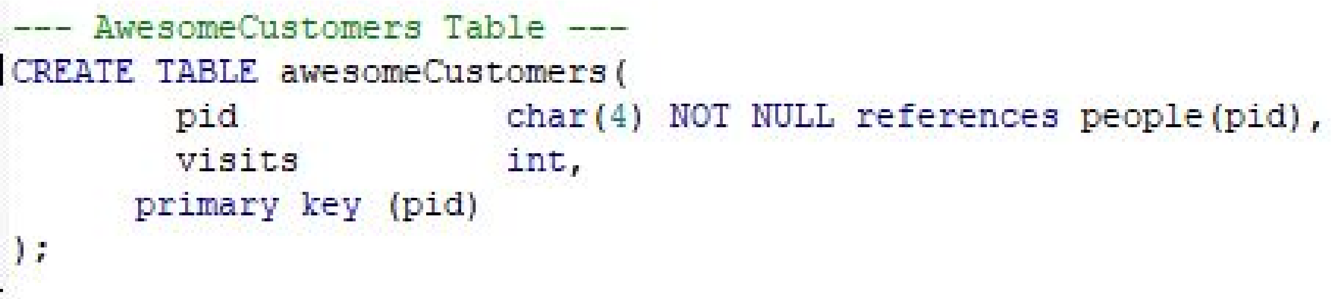
AwesomeCustomers consists of pid, dateOfFirstWash, and visits. This table may be used to calculate which customers visit frequently or have only visited once. It is important to remember that all customers are awesome.

Functional Dependency:

pid, dateOfFirstWash -> visits



	pid character(4)	visits integer
1	p001	23
2	p002	3
3	p003	13
4	p004	18
5	p005	36
6	p006	12
7	p007	27



```
--- AwesomeCustomers Table ---
CREATE TABLE awesomeCustomers(
        pid               char(4) NOT NULL references people(pid),
        visits            int,
      primary key (pid)
);
```

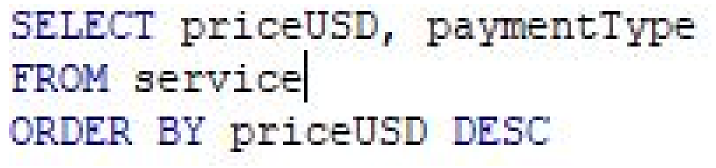
Check Constraint

I implemented a check constraint in the employees table to enforce that only yes or no can be inputted into the text field when asked if an employee has ever received a goldStar. The default value is no.

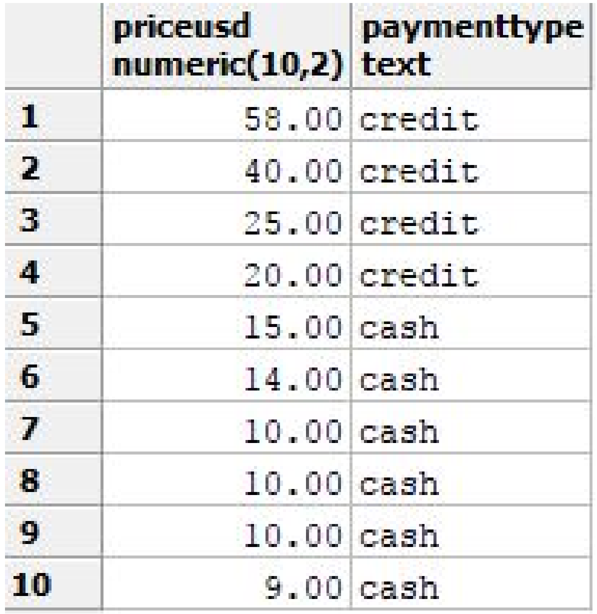
I also implemented a check constraint in the interior and exterior fields in the service table to enforce that only yes or no can be inputted into the text field. If there is no input the default is no.

Queries/Reports

This query is used to help identify whether there is a correlation between a particular payment type and the price.



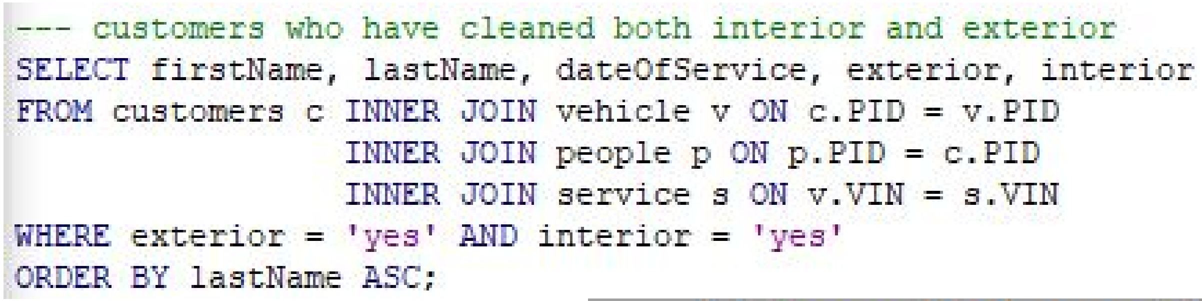
```
SELECT priceUSD, paymentType
FROM service
ORDER BY priceUSD DESC
```



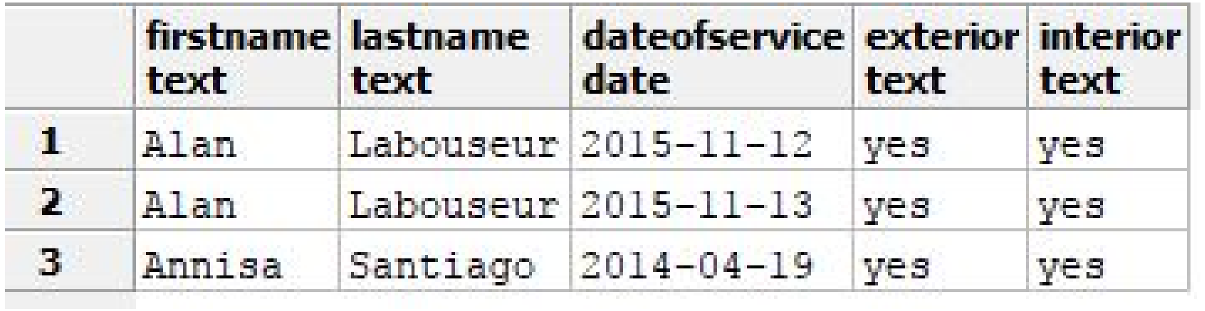
	priceusd numeric(10,2)	paymenttype text
1	58.00	credit
2	40.00	credit
3	25.00	credit
4	20.00	credit
5	15.00	cash
6	14.00	cash
7	10.00	cash
8	10.00	cash
9	10.00	cash
10	9.00	cash

Queries/Reports

Checks to see who and how many people clean both the interior and exterior of their vehicle. This could be useful when giving discounts to promote business.



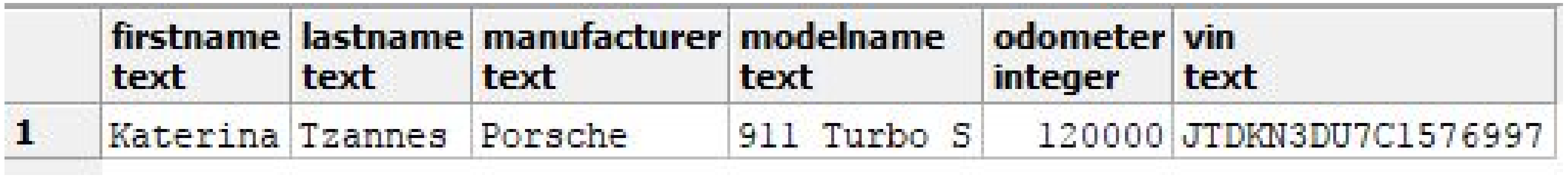
```
--- customers who have cleaned both interior and exterior
SELECT firstName, lastName, dateOfService, exterior, interior
FROM customers c INNER JOIN vehicle v ON c.PID = v.PID
                 INNER JOIN people p ON p.PID = c.PID
                 INNER JOIN service s ON v.VIN = s.VIN
WHERE exterior = 'yes' AND interior = 'yes'
ORDER BY lastName ASC;
```



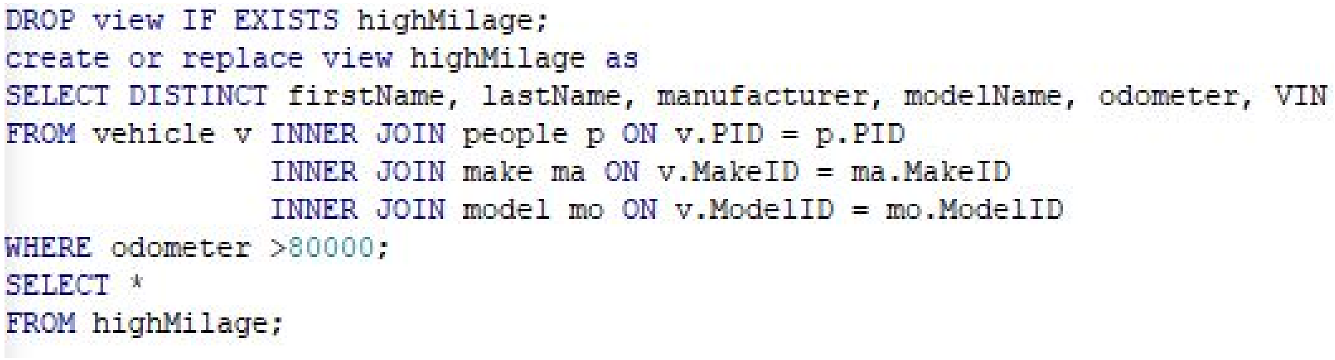
	firstname text	lastname text	dateofservice date	exterior text	interior text
1	Alan	Labouseur	2015-11-12	yes	yes
2	Alan	Labouseur	2015-11-13	yes	yes
3	Annisa	Santiago	2014-04-19	yes	yes

Views

This view shows individuals who have high milage on their vehicle.



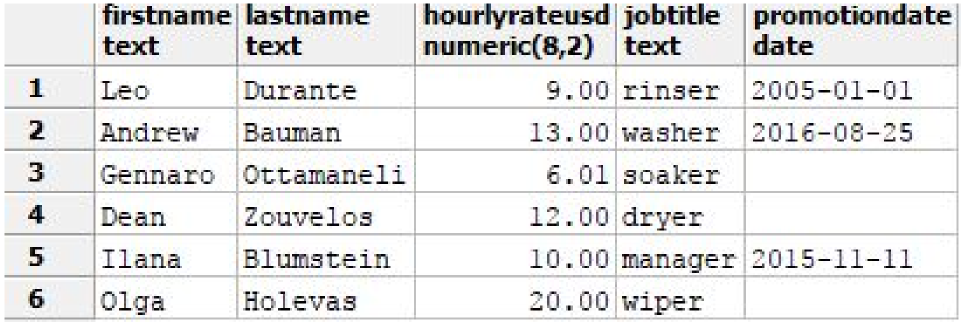
	firstname text	lastname text	manufacturer text	modelname text	odometer integer	vin text
1	Katerina	Tzannes	Porsche	911 Turbo S	120000	JTDKN3DU7C1576997



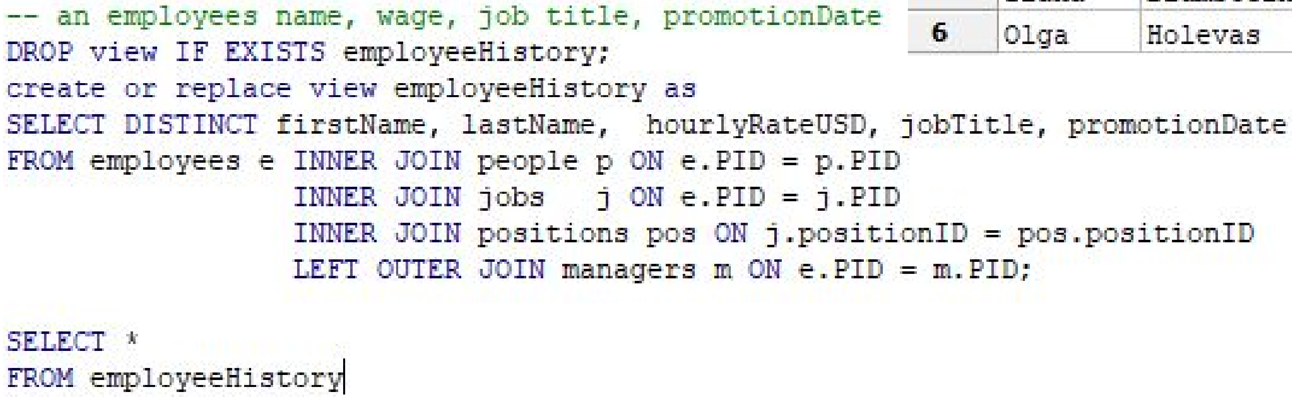
```
DROP view IF EXISTS highMilage;
create or replace view highMilage as
SELECT DISTINCT firstName, lastName, manufacturer, modelName, odometer, VIN
FROM vehicle v INNER JOIN people p ON v.PID = p.PID
               INNER JOIN make ma ON v.MakeID = ma.MakeID
               INNER JOIN model mo ON v.ModelID = mo.ModelID
WHERE odometer >80000;
SELECT *
FROM highMilage;
```

Views

This view shows an employees name, hourly rate, job title, and their promotion date if they are a manager.



	firstname text	lastname text	hourlyrateusd numeric(8,2)	jobtitle text	promotiondate date
1	Leo	Durante	9.00	rinser	2005-01-01
2	Andrew	Bauman	13.00	washer	2016-08-25
3	Gennaro	Ottamaneli	6.01	soaker	
4	Dean	Zouvelos	12.00	dryer	
5	Ilana	Blumstein	10.00	manager	2015-11-11
6	Olga	Holevas	20.00	wiper	



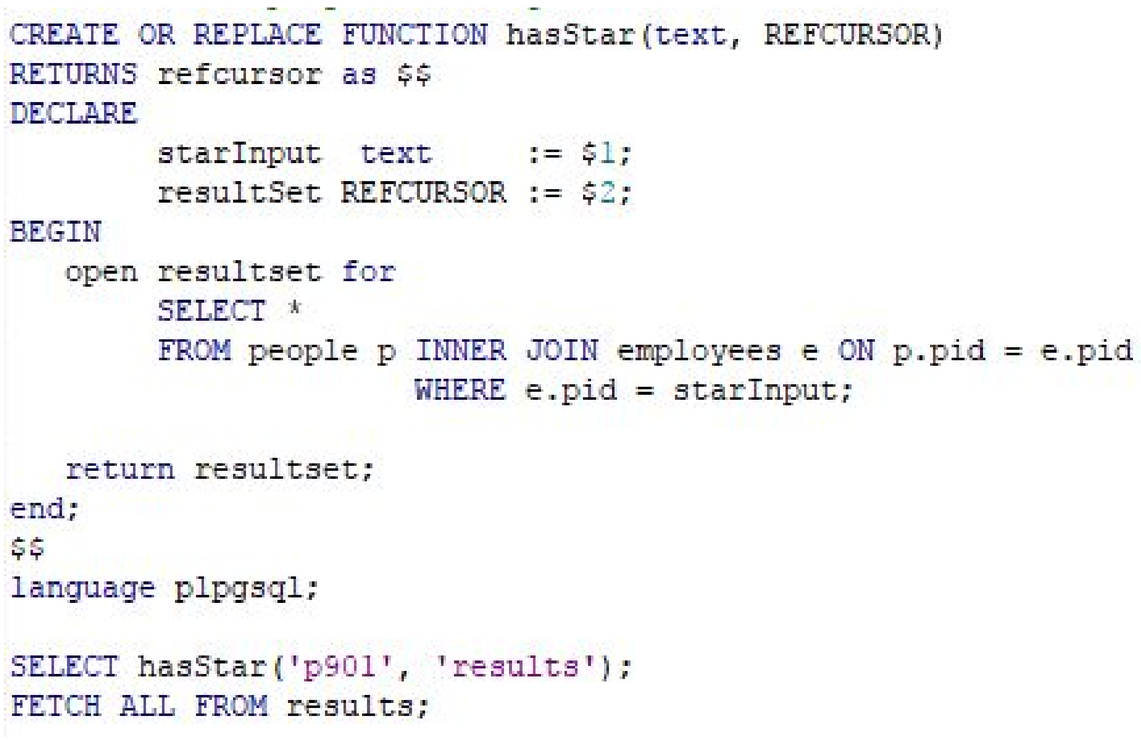
```
-- an employees name, wage, job title, promotionDate
DROP view IF EXISTS employeeHistory;
create or replace view employeeHistory as
SELECT DISTINCT firstName, lastName,  hourlyRateUSD, jobTitle, promotionDate
FROM employees e INNER JOIN people p ON e.PID = p.PID
                 INNER JOIN jobs   j ON e.PID = j.PID
                 INNER JOIN positions pos ON j.positionID = pos.positionID
                 LEFT OUTER JOIN managers m ON e.PID = m.PID;

SELECT *
FROM employeeHistory
```

Stored Procedures

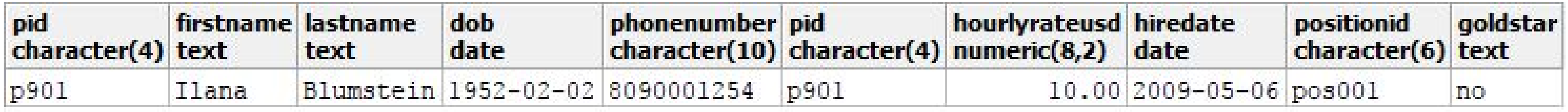
Returns if an

employee has a

gold star.



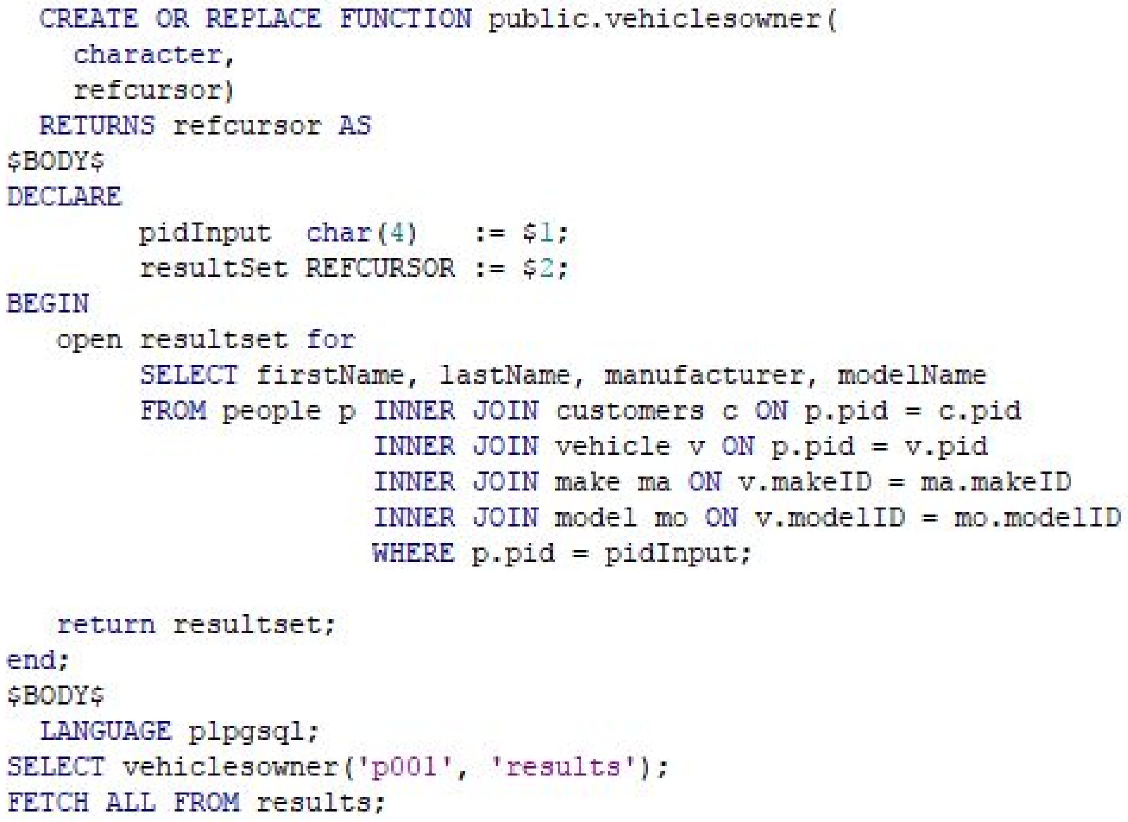
```
CREATE OR REPLACE FUNCTION hasStar(text, REFCURSOR)
RETURNS refcursor as $$
DECLARE
        starInput  text     := $1;
        resultSet REFCURSOR := $2;
BEGIN
   open resultset for
        SELECT *
        FROM people p INNER JOIN employees e ON p.pid = e.pid
              WHERE e.pid = starInput;

   return resultset;
end;
$$
language plpgsql;

SELECT hasStar('p901', 'results');
FETCH ALL FROM results;
```



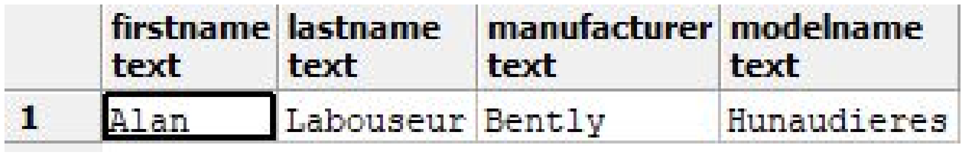
pid character(4)	firstname text	lastname text	dob date	phonenumber character(10)	pid character(4)	hourlyrateusd numeric(8,2)	hiredate date	positionid character(6)	goldstar text
p901	Ilana	Blumstein	1952-02-02	8090001254	p901	10.00	2009-05-06	pos001	no

Stored Procedures

Returns who owns which vehicle



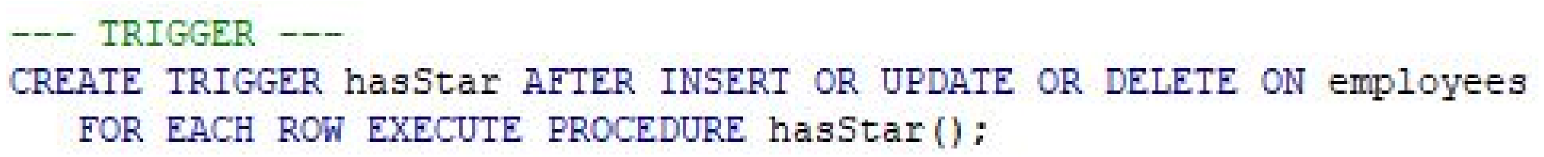
```
  CREATE OR REPLACE FUNCTION public.vehiclesowner(
    character,
    refcursor)
  RETURNS refcursor AS
$BODY$
DECLARE
        pidInput  char(4)   := $1;
        resultSet REFCURSOR := $2;
BEGIN
   open resultset for
        SELECT firstName, lastName, manufacturer, modelName
        FROM people p INNER JOIN customers c ON p.pid = c.pid
                      INNER JOIN vehicle v ON p.pid = v.pid
                      INNER JOIN make ma ON v.makeID = ma.makeID
                      INNER JOIN model mo ON v.modelID = mo.modelID
                      WHERE p.pid = pidInput;

   return resultset;
end;
$BODY$
  LANGUAGE plpgsql;
SELECT vehiclesowner('p001', 'results');
FETCH ALL FROM results;
```



	firstname text	lastname text	manufacturer text	modelname text
1	Alan	Labouseur	Bently	Hunaudieres

Triggers

This trigger is linked with the hasStar store procedure. It is intended to prevent employees from having an incorrect amount of stars.



```
--- TRIGGER ---
CREATE TRIGGER hasStar AFTER INSERT OR UPDATE OR DELETE ON employees
   FOR EACH ROW EXECUTE PROCEDURE hasStar();
```

Known Problems

Connecting the awesomeCustomers table with the Service table would help properly reflect the true number of customer visits and help keep the database normalized.

Future Enhancements

A future enhancement could be storing the amount of products, like soap. Storing products used could help keep track of how much stock is left. This would be helpful in order to predict when it is necessary to reorder products and estimate how long they should last.