Moto Gear

DESIGN PROJECT

Dan Swezey
CMPT 308

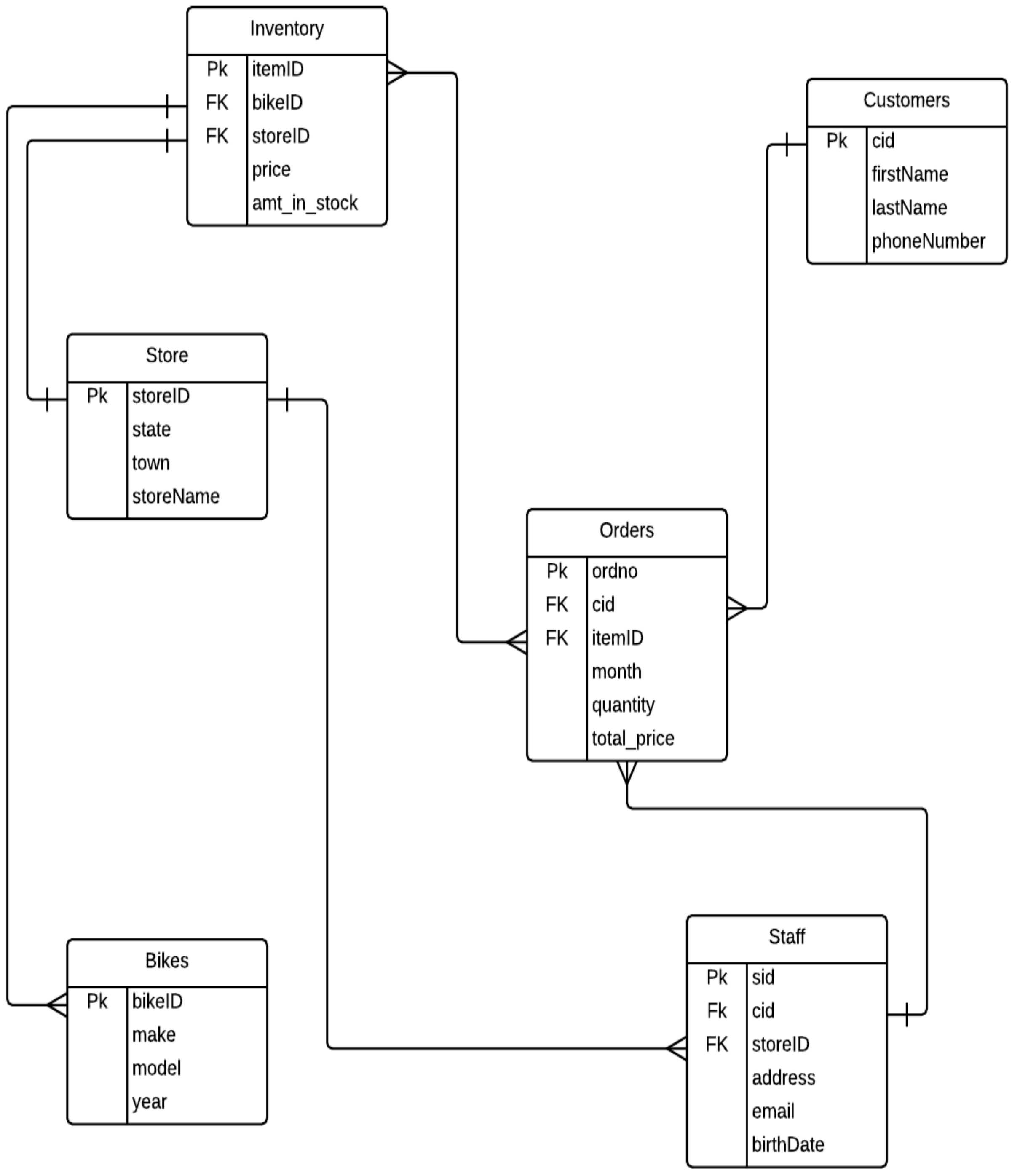
Table of Content

Executive Summary

What we have here is a well-structured database for a Motorcycle company called *Moto Gear*. It will be used to view all of the information that the company needs to know. This includes customers the company has, different store locations, information on staff, what kind of bikes are carried, inventory for each store, and even individual orders/purchases made from customers. This would solely be used for administration and staff.

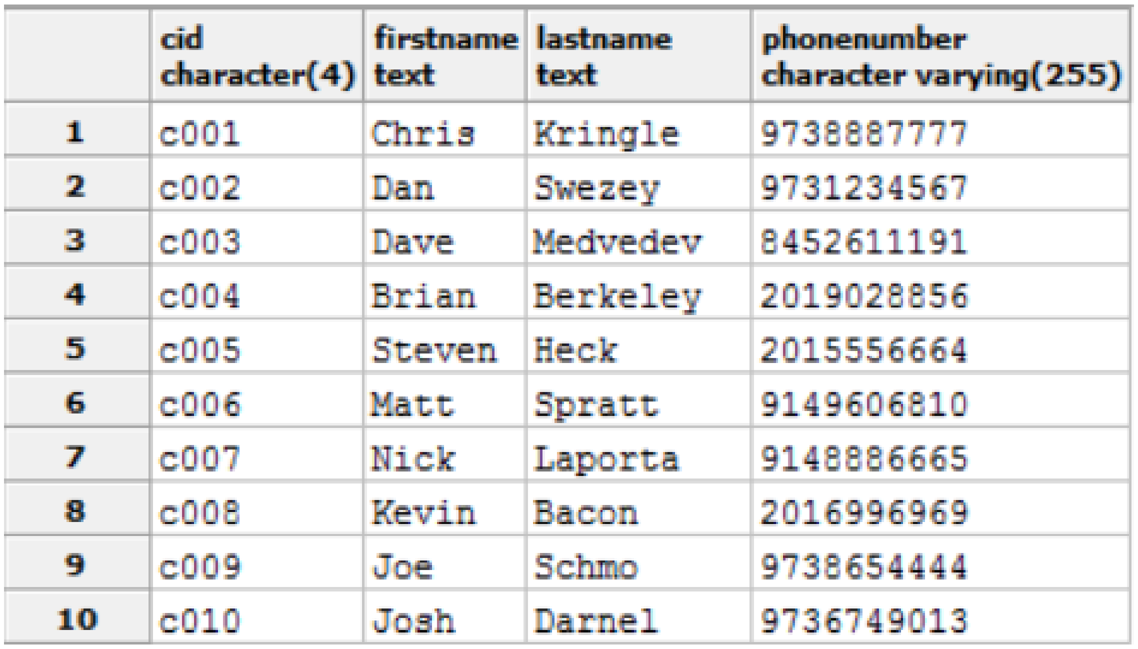
Throughout this document, there will be the Entity Relationship Diagram showing the relationships between the tables in the database. The tables themselves along with the SQL code for the tables, functional dependencies, and some sample data are a part of this document. Additional coding and sample data is shown for the views, reports, and stored procedures. There are also some security roles showing the restrictions in the database. The documentation is concluded with some notes and known issues/enhancements that could be added or removed from the database.

Entity Relationship Diagram



Tables

Customers Table



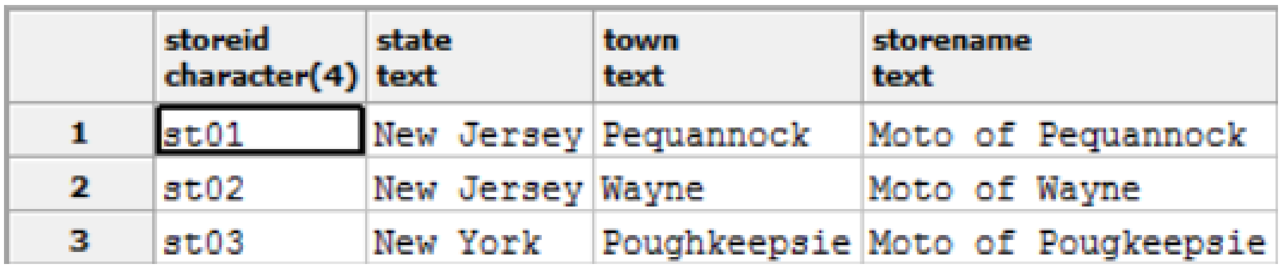
	cid character(4)	firstname text	lastname text	phonenumber character varying(255)
1	c001	Chris	Kringle	9738887777
2	c002	Dan	Swezey	9731234567
3	c003	Dave	Medvedev	8452611191
4	c004	Brian	Berkeley	2019028856
5	c005	Steven	Heck	2015556664
6	c006	Matt	Spratt	9149606810
7	c007	Nick	Laporta	9148886665
8	c008	Kevin	Bacon	2016996969
9	c009	Joe	Schmo	9738654444
10	c010	Josh	Darnel	9736749013

```
CREATE TABLE customers (
cid char(4) not null,
firstName   text,
lastName    text,
phoneNumber varchar(255),
primary key(cid)
);
```

Functional Dependencies: cid -> firstName, lastName, phoneNumber

- This table contains a list of all customers who have bought something this store. It includes the customer id, first name, last name, and their phone number.

<u>Store Table</u>



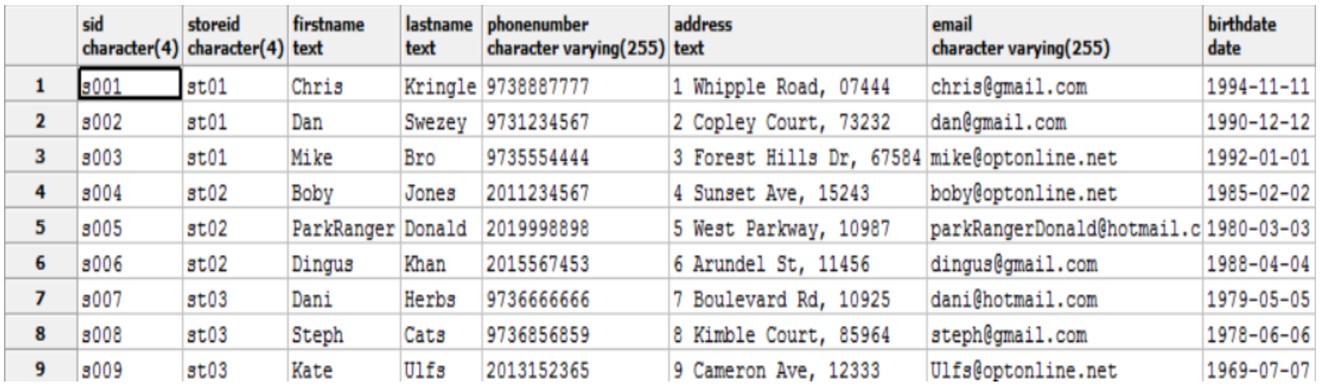
	storeid character(4)	state text	town text	storename text
1	st01	New Jersey	Pequannock	Moto of Pequannock
2	st02	New Jersey	Wayne	Moto of Wayne
3	st03	New York	Poughkeepsie	Moto of Pougkeepsie

```
CREATE TABLE store (
storeID char(4) not null,
state text,
town text,
storeName text,
primary key(storeID)
);
```

Functional Dependencies: storeID -> state, town, storeName

- This table is a list of the companies' stores. It includes the store id, state and in which the store is located, and the store name.

Staff Table



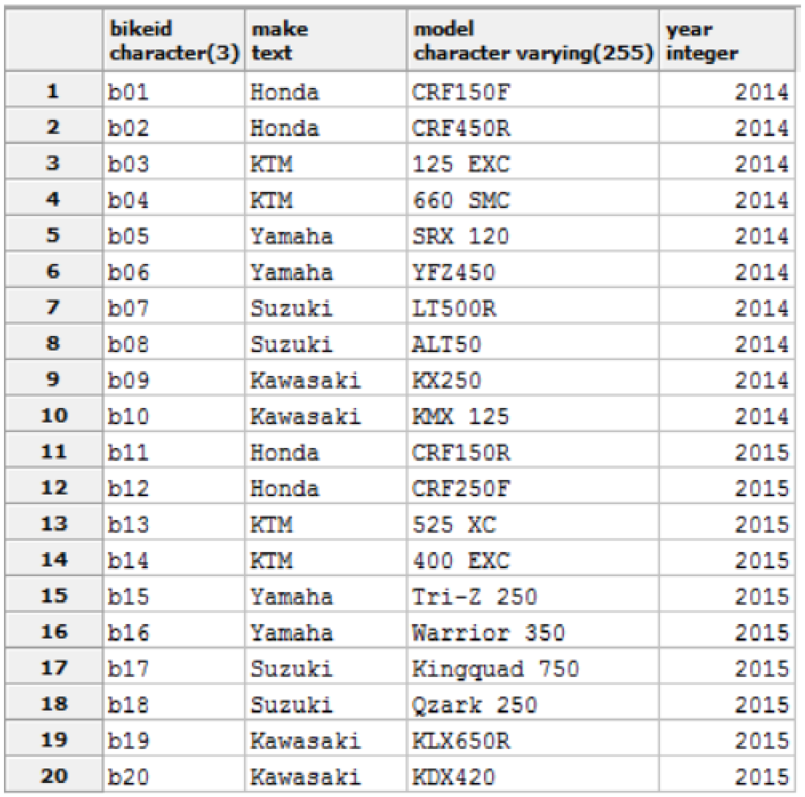
	sid character(4)	storeid character(4)	firstname text	lastname text	phonenumber character varying(255)	address text	email character varying(255)	birthdate date
1	s001	st01	Chris	Kringle	9738887777	1 Whipple Road, 07444	chris@gmail.com	1994-11-11
2	s002	st01	Dan	Swezey	9731234567	2 Copley Court, 73232	dan@gmail.com	1990-12-12
3	s003	st01	Mike	Bro	9735554444	3 Forest Hills Dr, 67584	mike@optonline.net	1992-01-01
4	s004	st02	Boby	Jones	2011234567	4 Sunset Ave, 15243	boby@optonline.net	1985-02-02
5	s005	st02	ParkRanger	Donald	2019998898	5 West Parkway, 10987	parkRangerDonald@hotmail.c	1980-03-03
6	s006	st02	Dingus	Khan	2015567453	6 Arundel St, 11456	dingus@gmail.com	1988-04-04
7	s007	st03	Dani	Herbs	9736666666	7 Boulevard Rd, 10925	dani@hotmail.com	1979-05-05
8	s008	st03	Steph	Cats	9736856859	8 Kimble Court, 85964	steph@gmail.com	1978-06-06
9	s009	st03	Kate	Ulfs	2013152365	9 Cameron Ave, 12333	Ulfs@optonline.net	1969-07-07

```
CREATE TABLE staff (
sid char(4) not null,
storeID char(4) not null REFERENCES store(storeID),
firstName text,
lastName text,
phoneNumber varchar(255),
address text,
email varchar(255),
birthDate date,
primary key(sid)
);
```

Functional Dependencies: sid -> address, birthdate, email

- This table is a list of all staff members throughout the company. It shows the staff id, store id in which they work, first name, last name, phone number, address, email, and date of birth.

<u>Bikes Table</u>



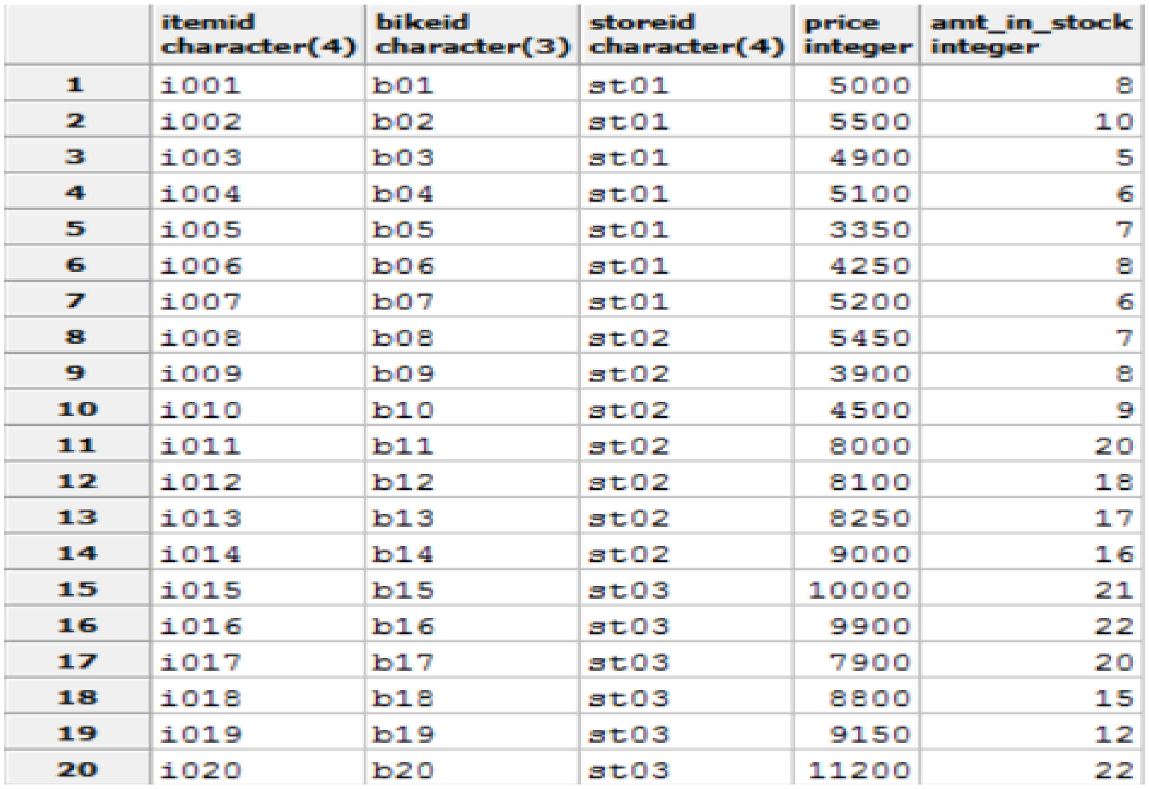
	bikeid character(3)	make text	model character varying(255)	year integer
1	b01	Honda	CRF150F	2014
2	b02	Honda	CRF450R	2014
3	b03	KTM	125 EXC	2014
4	b04	KTM	660 SMC	2014
5	b05	Yamaha	SRX 120	2014
6	b06	Yamaha	YFZ450	2014
7	b07	Suzuki	LT500R	2014
8	b08	Suzuki	ALT50	2014
9	b09	Kawasaki	KX250	2014
10	b10	Kawasaki	KMX 125	2014
11	b11	Honda	CRF150R	2015
12	b12	Honda	CRF250F	2015
13	b13	KTM	525 XC	2015
14	b14	KTM	400 EXC	2015
15	b15	Yamaha	Tri-Z 250	2015
16	b16	Yamaha	Warrior 350	2015
17	b17	Suzuki	Kingquad 750	2015
18	b18	Suzuki	Qzark 250	2015
19	b19	Kawasaki	KLX650R	2015
20	b20	Kawasaki	KDX420	2015

CREATE TABLE bikes (
bikeID char(3) not null,
make text,
model varchar(255),
year int,
primary key(bikeID)
);

Functional Dependencies: bikeID -> make, model, year

- This table is a list of all the bikes the company sells. It includes the bike id, make and model of the bike and what year it was made.

Inventory Table



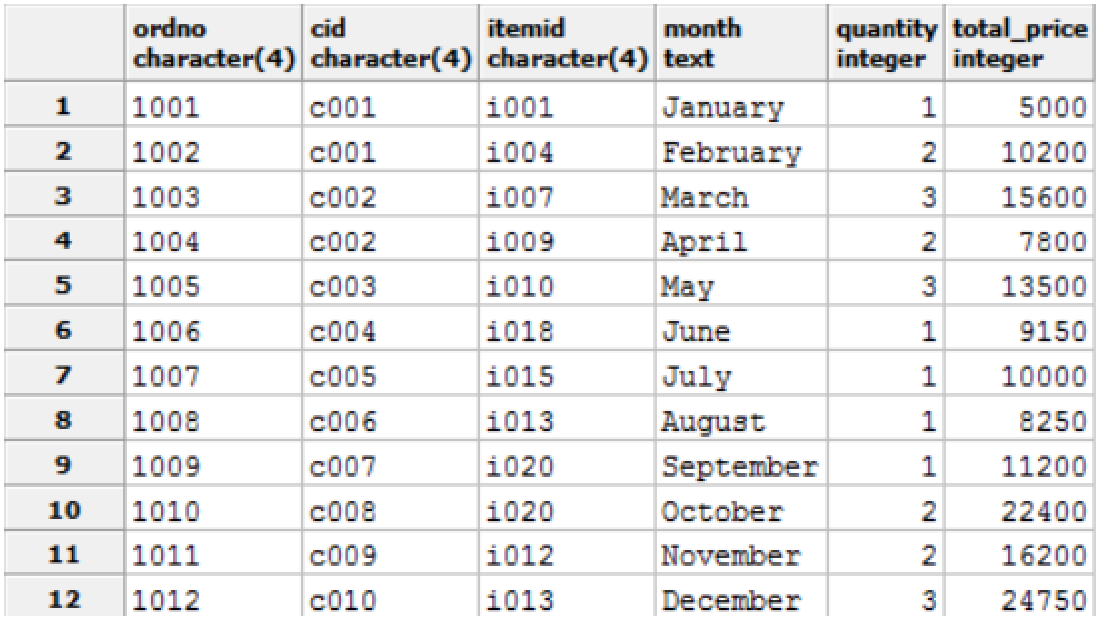
	itemid character(4)	bikeid character(3)	storeid character(4)	price integer	amt_in_stock integer
1	i001	b01	st01	5000	8
2	i002	b02	st01	5500	10
3	i003	b03	st01	4900	5
4	i004	b04	st01	5100	6
5	i005	b05	st01	3350	7
6	i006	b06	st01	4250	8
7	i007	b07	st01	5200	6
8	i008	b08	st02	5450	7
9	i009	b09	st02	3900	8
10	i010	b10	st02	4500	9
11	i011	b11	st02	8000	20
12	i012	b12	st02	8100	18
13	i013	b13	st02	8250	17
14	i014	b14	st02	9000	16
15	i015	b15	st03	10000	21
16	i016	b16	st03	9900	22
17	i017	b17	st03	7900	20
18	i018	b18	st03	8800	15
19	i019	b19	st03	9150	12
20	i020	b20	st03	11200	22

```
CREATE TABLE inventory (
itemID char(4) not null,
bikeID char(3) not null REFERENCES bikes(bikeID),
storeID char(4) not null REFERENCES store(storeID),
price int,
amt_in_stock int,
primary key(itemID)
);
```

Functional Dependencies: itemID -> price, amt_in_stock

- This table shows what is actually in stock in each store. It includes an item id, the bike id, the stores id, price of the bike, and how many are in stock.

<u>Orders Table</u>



	ordno	cid	itemid	month	quantity	total_price
	character(4)	character(4)	character(4)	text	integer	integer
1	1001	c001	i001	January	1	5000
2	1002	c001	i004	February	2	10200
3	1003	c002	i007	March	3	15600
4	1004	c002	i009	April	2	7800
5	1005	c003	i010	May	3	13500
6	1006	c004	i018	June	1	9150
7	1007	c005	i015	July	1	10000
8	1008	c006	i013	August	1	8250
9	1009	c007	i020	September	1	11200
10	1010	c008	i020	October	2	22400
11	1011	c009	i012	November	2	16200
12	1012	c010	i013	December	3	24750

```
CREATE TABLE orders (
ordno char(4) not null,
cid char(4) not null REFERENCES customers(cid),
itemID char(4) not null REFERENCES inventory(itemID),
month text,
quantity int,
total_price int,
primary key(ordno)
);
```

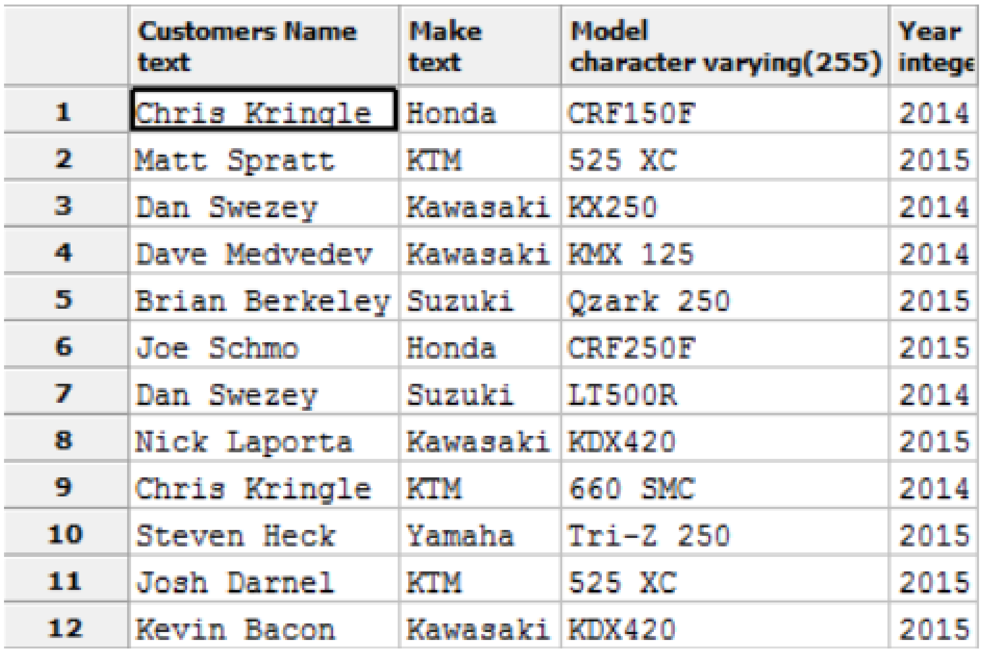
Fuctional Dependencies: ordno -> month, quantity, total_price

- This table shows orders made by customers. It has the order number, customer id, item id, month ordered, how many items were ordered, and total price of the order.

Views

- This view shows customers and what they ordered.

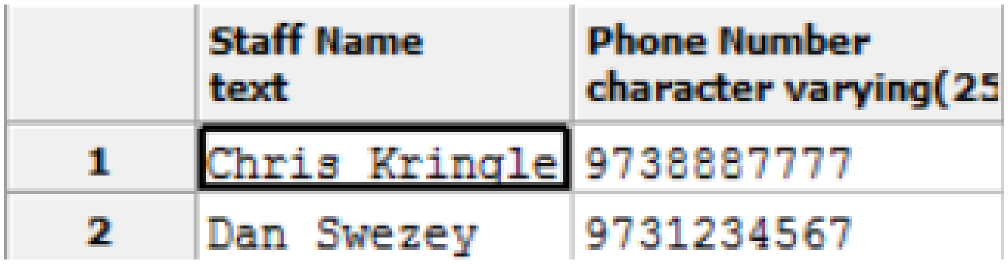
```
CREATE VIEW CustomersOrders AS
SELECT DISTINCT customers.firstName||' '||customers.lastName as "Customers
Name",
bikes.make as "Make", bikes.model as "Model", bikes.year as "Year"
FROM customers
INNER JOIN orders ON orders.cid = customers.cid
INNER JOIN inventory ON orders.itemID = inventory.itemID
INNER JOIN bikes ON inventory.bikeID = bikes.bikeID
```



	Customers Name text	Make text	Model character varying(255)	Year intege
1	Chris Kringle	Honda	CRF150F	2014
2	Matt Spratt	KTM	525 XC	2015
3	Dan Swezey	Kawasaki	KX250	2014
4	Dave Medvedev	Kawasaki	KMX 125	2014
5	Brian Berkeley	Suzuki	Ozark 250	2015
6	Joe Schmo	Honda	CRF250F	2015
7	Dan Swezey	Suzuki	LT500R	2014
8	Nick Laporta	Kawasaki	KDX420	2015
9	Chris Kringle	KTM	660 SMC	2014
10	Steven Heck	Yamaha	Tri-Z 250	2015
11	Josh Darnel	KTM	525 XC	2015
12	Kevin Bacon	Kawasaki	KDX420	2015

- This view shows staff members that are also customers

```
CREATE VIEW StaffCustomers AS
SELECT DISTINCT staff.firstName||' '||staff.lastName as "Staff Name",
staff.phoneNumber as "Phone Number"
FROM staff
INNER JOIN customers ON customers.firstName = staff.firstName AND
customers.lastName = staff.lastName AND
customers.phoneNumber = staff.phoneNumber
```

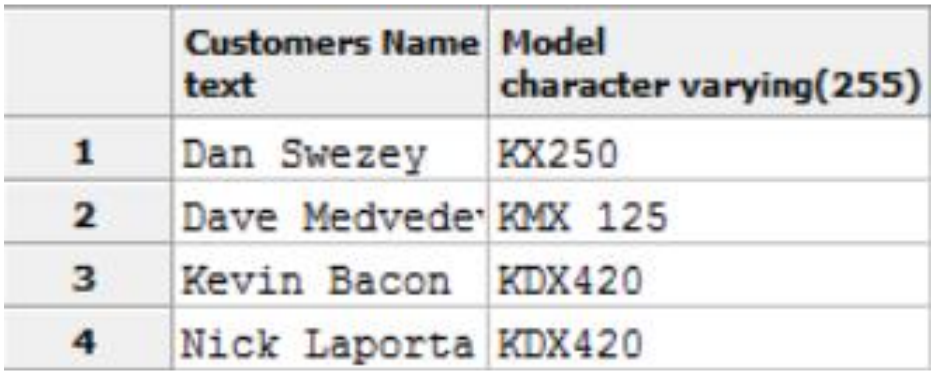


	Staff Name text	Phone Number character varying(25
1	Chris Kringle	9738887777
2	Dan Swezey	9731234567

Reports

- This report shows the customers who ordered a Kawasaki

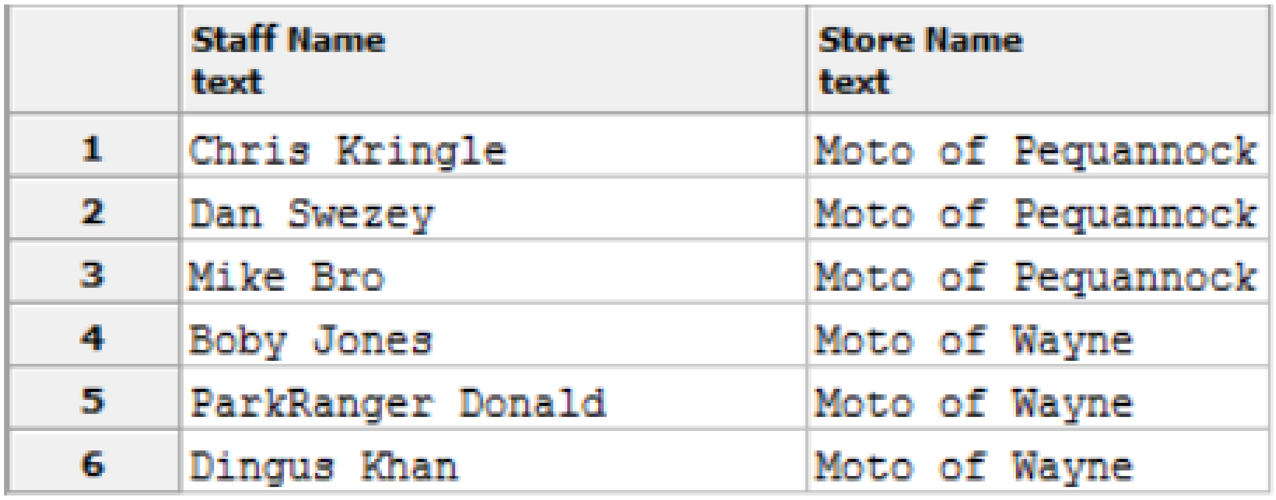
```
SELECT "Customers Name", "Model"
FROM CustomersOrders
WHERE "Make" = 'Kawasaki'
```



	Customers Name text	Model character varying(255)
1	Dan Swezey	KX250
2	Dave Medvede	KMX 125
3	Kevin Bacon	KDX420
4	Nick Laporta	KDX420

- This report shows the staff members that work in New Jersey

```
SELECT staff.firstName||' '||staff.lastName as "Staff Name", store.storeName as
"Store Name"
FROM staff INNER JOIN store ON store.storeID = staff.storeID
WHERE store.state = 'New Jersey'
```



	Staff Name text	Store Name text
1	Chris Kringle	Moto of Pequannock
2	Dan Swezey	Moto of Pequannock
3	Mike Bro	Moto of Pequannock
4	Boby Jones	Moto of Wayne
5	ParkRanger Donald	Moto of Wayne
6	Dingus Khan	Moto of Wayne

Stored Procedures

- Select the make of a bike. Have it return the make, model, and year of specified make.

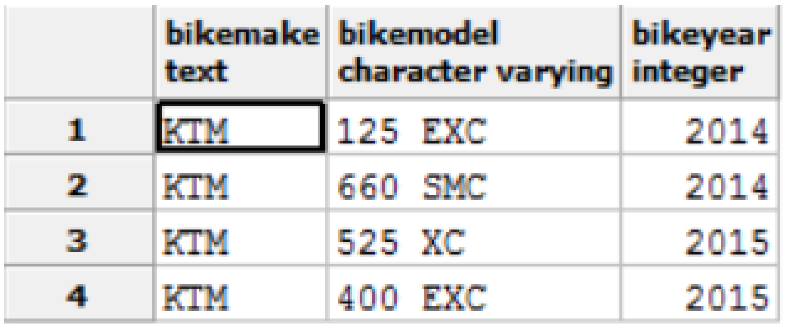
```
CREATE FUNCTION getSpecs(brand TEXT, OUT bikeMake TEXT, OUT bikeModel
VARCHAR, OUT bikeYear INT)
RETURNS SETOF RECORD AS $$

BEGIN
RETURN QUERY select make, model, year
FROM bikes
WHERE bikes.make = brand;
END;

$$ LANGUAGE plpgsql;
```

EX:

```
select * from getSpecs('KTM');
```



	bikemake text	bikemodel character varying	bikeyear integer
1	KTM	125 EXC	2014
2	KTM	660 SMC	2014
3	KTM	525 XC	2015
4	KTM	400 EXC	2015

- Select a quantity of an order. Return that quantity, the order number, and the month the order was made.

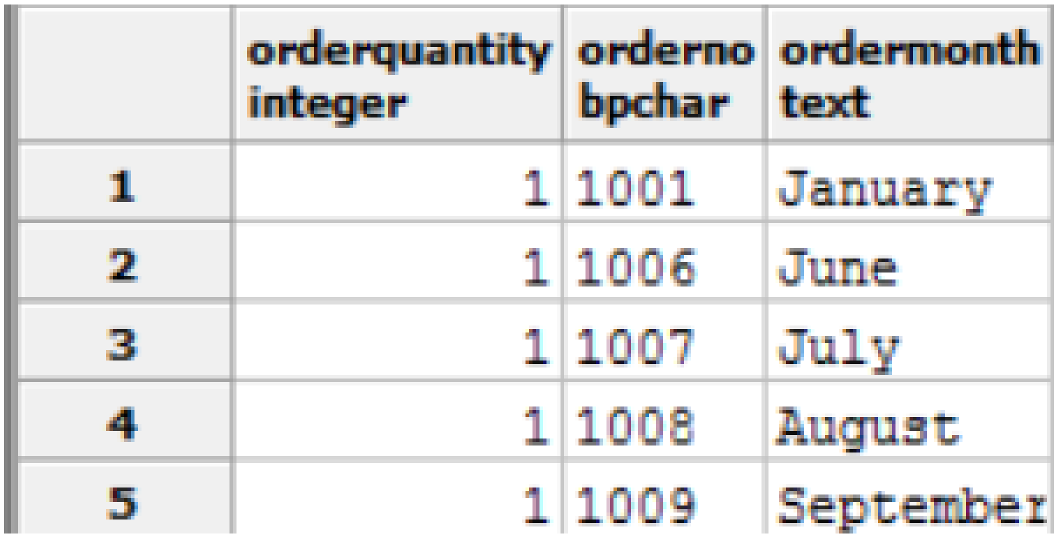
```
CREATE FUNCTION getQua(number INT, OUT orderQuantity INT, OUT orderNo
CHAR, OUT orderMonth TEXT)
RETURNS SETOF RECORD AS $$

BEGIN
RETURN QUERY select orders.quantity, orders.ordno, orders.month
FROM orders
WHERE orders.quantity = number;
END;

$$ LANGUAGE plpgsql;
```

EX:

```
select * from getQua(1);
```

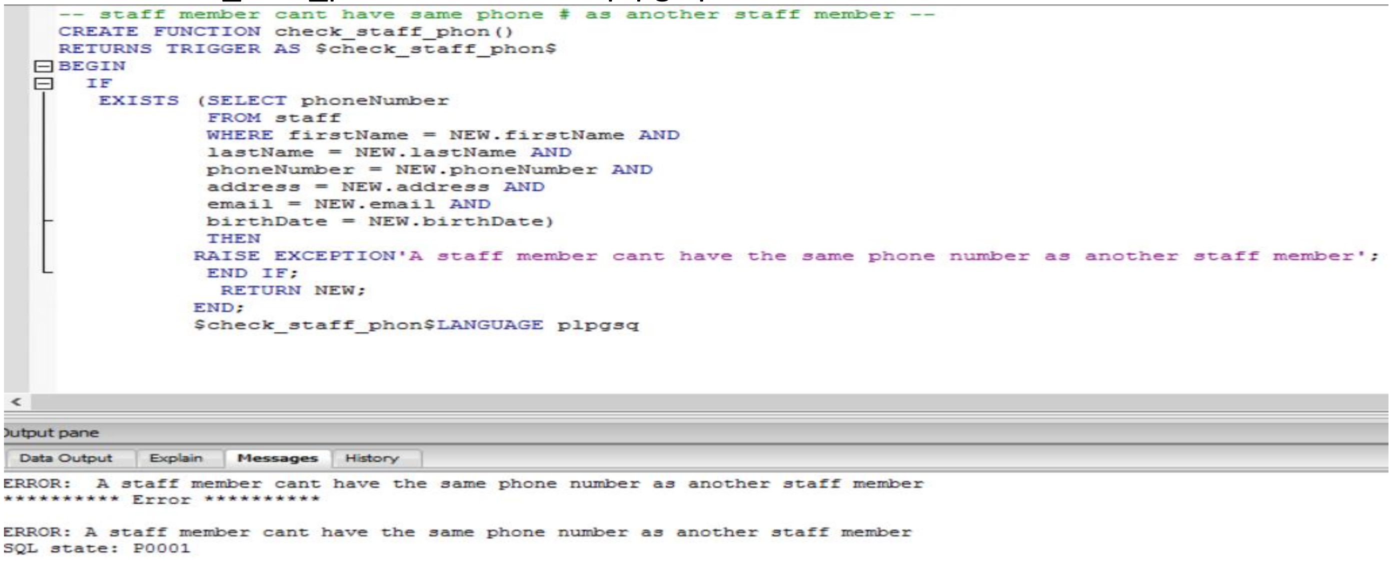


	orderquantity integer	orderno bpchar	ordermonth text
1	1	1001	January
2	1	1006	June
3	1	1007	July
4	1	1008	August
5	1	1009	September

Triggers

- A staff member cannot have the same phone number as another staff member.

```
CREATE FUNCTION check_staff_phon()
RETURNS TRIGGER AS $check_staff_phon$
BEGIN
 IF
  EXISTS (SELECT phoneNumber
          FROM staff
          WHERE firstName = NEW.firstName AND
          lastName = NEW.lastName AND
          phoneNumber = NEW.phoneNumber AND
          address = NEW.address AND
          email = NEW.email AND
          birthDate = NEW.birthDate)
          THEN
         RAISE EXCEPTION'A staff member cant have the same phone number as
another staff member';
          END IF;
           RETURN NEW;
         END;
         $check_staff_phon$LANGUAGE plpgsq
```



```
-- staff member cant have same phone # as another staff member --
CREATE FUNCTION check_staff_phon()
RETURNS TRIGGER AS $check_staff_phon$
BEGIN
  IF
   EXISTS (SELECT phoneNumber
            FROM staff
            WHERE firstName = NEW.firstName AND
            lastName = NEW.lastName AND
            phoneNumber = NEW.phoneNumber AND
            address = NEW.address AND
            email = NEW.email AND
            birthDate = NEW.birthDate)
            THEN
           RAISE EXCEPTION'A staff member cant have the same phone number as another staff member';
            END IF;
             RETURN NEW;
           END;
           $check_staff_phon$LANGUAGE plpgsq
```

```
ERROR:  A staff member cant have the same phone number as another staff member
********** Error **********

ERROR: A staff member cant have the same phone number as another staff member
SQL state: P0001
```

Check ^

Security

- The first type of security would be for an admin. They can insert, update, and alter the database.

- CREATE ROLE admin WITH CREATEDB CREATEROLE;

- The second type of security would be for the staff. They can only see and make queries on the database.

- CREATE ROLE staff WITH CREATEDB;

Implementation notes
Known Problems
Future Enhancements

When it came to the implementation, there were very small things that slowed down the process. Some of them were small syntax errors or missing semi-colons, etc. The only way to fix those issues is to carefully scan and revise the code. Another step to avoid issues was to keep things simple, but still have all the essentials. To do this, it is important that the tables are not overly complicated and the ER Diagram is neat and easy to read.

A known problem is that phone numbers of customers entered into the database cannot be the same, which means there is the possibility for redundant data. The way to fix that would be with a future enhancement by making a trigger to prevent it.

Any other future enhancement would be based off of how the company changes. This could mean adding tables or reformatting data.