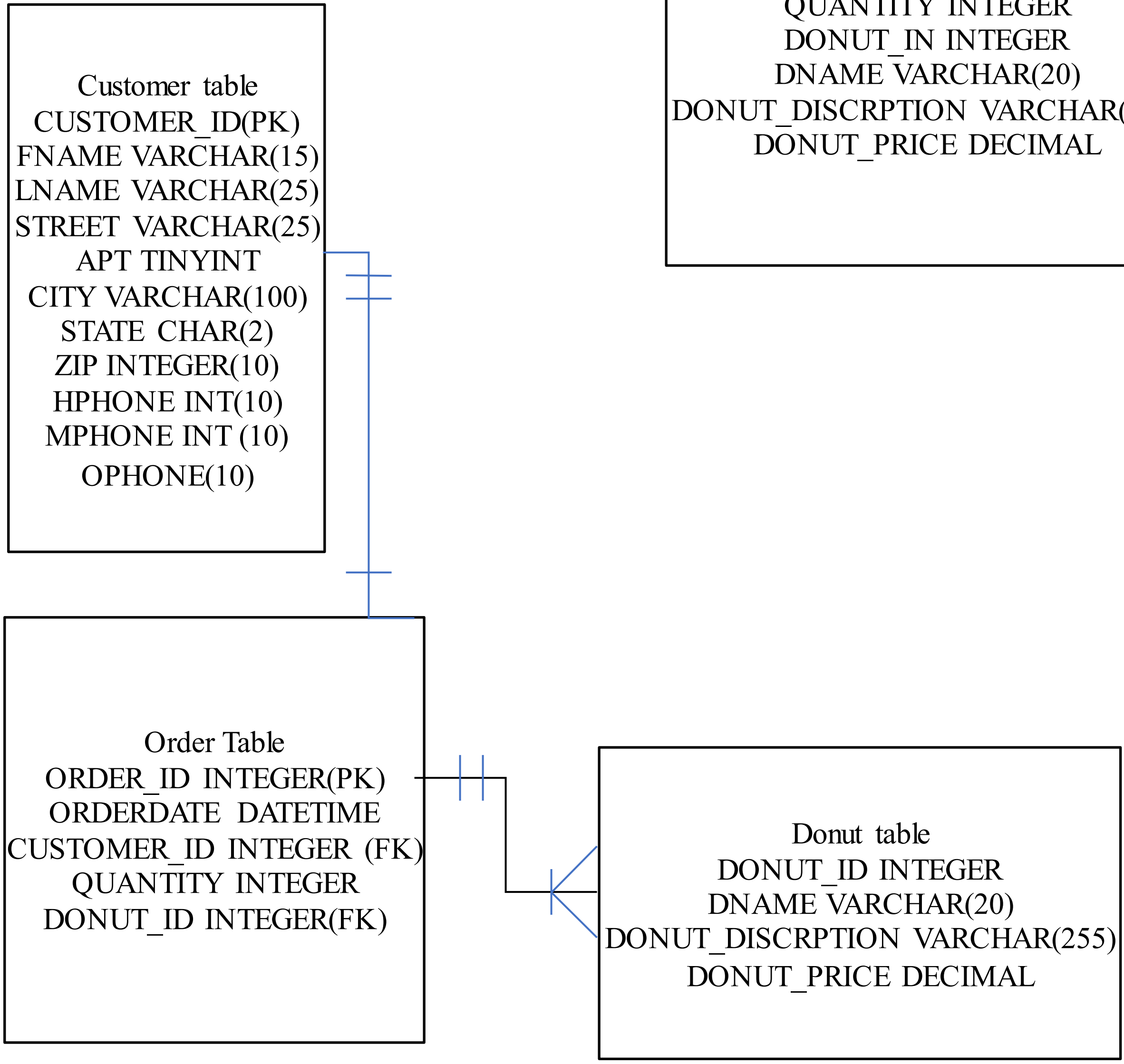
WGU Database Management Applications Project

Starting with the sales order invoice we take the data for customer name, address, and the order containing the donuts, description, and price. After the data is atomized we are left with a table in first normal form. Looking at the first table we remove and entries that are not dependent on both of the primary keys we than end up with three tables in second normal form.

FNAME VARCHAR(15)
LNAME VARCHAR(25)
STREET VARCHAR(25)
APT TINYINT
CITY VARCHAR(100)
STATE CHAR(2)
ZIP INTEGER(10)
HPHONE INT(10)
MPHONE INT (10)
OPHONE(10)
ORDER_ID INTEGER(PK)
ORDERDATE DATETIME
CUSTOMER_ID INTEGER (PK)
QUANTITY INTEGER
DONUT_IN INTEGER
DNAME VARCHAR(20)
DONUT_DISCRPTION VARCHAR(255)
DONUT_PRICE DECIMAL

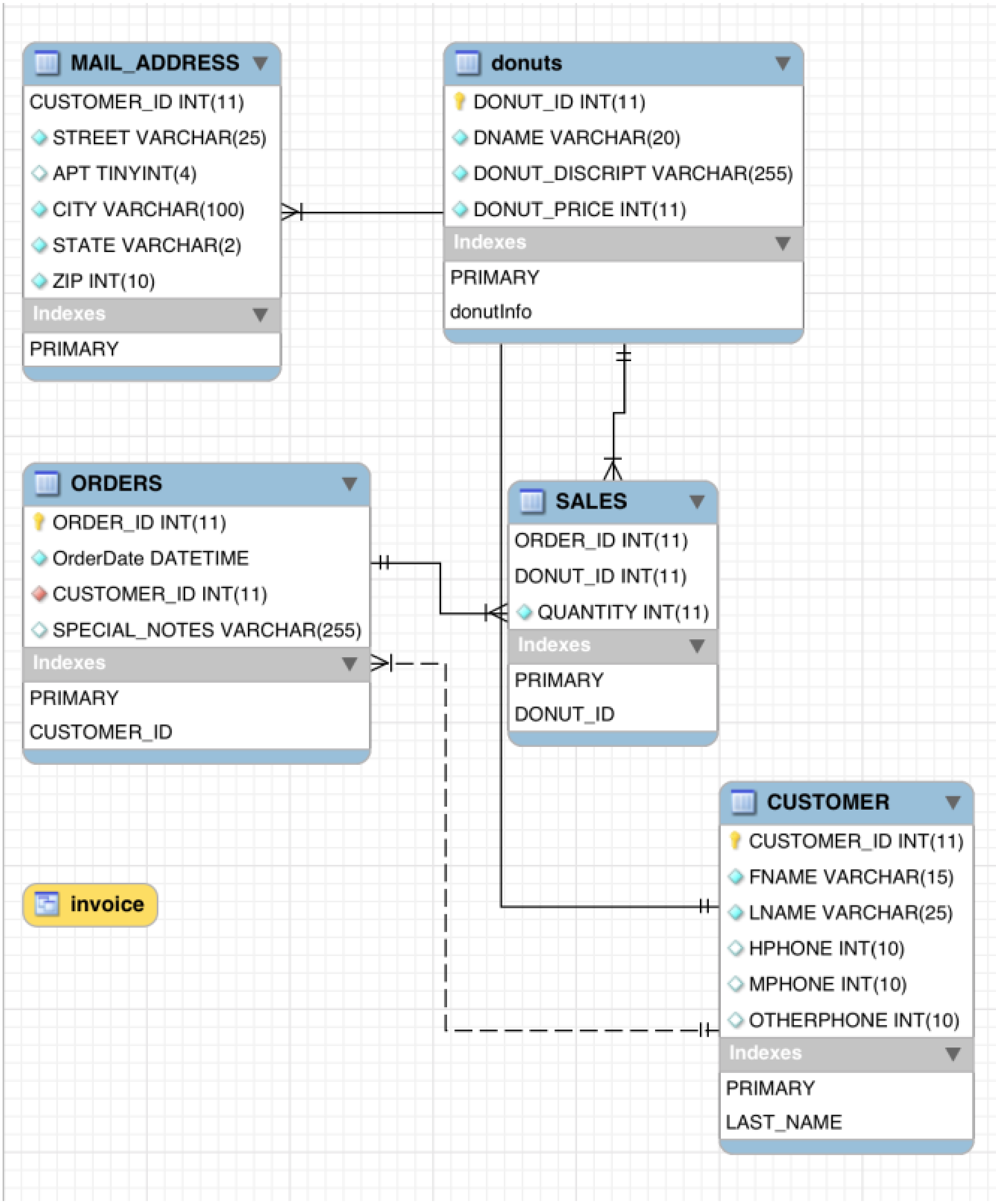


Customer table
CUSTOMER_ID(PK)
FNAME VARCHAR(15)
LNAME VARCHAR(25)
STREET VARCHAR(25)
APT TINYINT
CITY VARCHAR(100)
STATE CHAR(2)
ZIP INTEGER(10)
HPHONE INT(10)
MPHONE INT (10)
OPHONE(10)

Order Table
ORDER_ID INTEGER(PK)
ORDERDATE DATETIME
CUSTOMER_ID INTEGER (FK)
QUANTITY INTEGER
DONUT_ID INTEGER(FK)

Donut table
DONUT_ID INTEGER
DNAME VARCHAR(20)
DONUT_DISCRPTION VARCHAR(255)
DONUT_PRICE DECIMAL

With the data in the second normal form we further isolate the data to limit duplicate data and repeating groups. Creating a table for the mail addresses from customer table so there is no transitive dependencies. Next move the move the order ids and special notes and order date from sales table into a orders table which then makes the sales table just the stored entries of sold donuts identified by a order id as a foreign key. Bellow you find an entity relationship diagram made with MySQL workbench model application. All tables have been taken to the third normal form. The customer table is in a one to n relation with mail_address because a customer could move and need a new address. The two table were broken apart to remove the transitive relationship between the address and the customer_id(pk). The mail_address hold the customer_id as a primary key. The customer table is then linked through a foreign key to the orders table. The orders table is links the customer with sales table. The sales table generates a order_id gives the order a time stamp and connects that to the customer_id. The sales table takes the order_id and donut_id as a foreign keys that are a composite primary key to record who is buying which donut and how many. Which brings us to the donut table which stores the donut data. Name, description and price.



Below is the sql ddl code to create the data base, tables, view and index of the exercise.

CREATE DATABASE PROJECTDATABASE;

```
CREATE TABLE CUSTOMER (
   CUSTOMER_ID INTEGER AUTO_INCREMENT PRIMARY KEY,
   FNAME VARCHAR(15) NOT NULL,
   LNAME VARCHAR(25) NOT NULL,
   HPHONE INT(10),
   MPHONE INT(10),
   OTHERPHONE INT(10));
CREATE TABLE MAIL_ADDRESS(
   CUSTOMER_ID INTEGER AUTO_INCREMENT PRIMARY KEY,
   STREET VARCHAR (25) NOT NULL,
  APT TINYINT,
   CITY VARCHAR (100) NOT NULL,
   STATE VARCHAR(2) NOT NULL,
   ZIP INTEGER(10) NOT NULL,
   FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID)
);
CREATE TABLE DONUTS(
   DONUT_ID INTEGER AUTO_INCREMENT PRIMARY KEY,
   DNAME VARCHAR(20) NOT NULL,
   DONUT_DESCRIPT VARCHAR(255) NOT NULL,
   DONUT_PRICE DEC(3,2) NOT NULL);
  CREATE TABLE ORDERS (
   ORDER_ID INTEGER AUTO_INCREMENT PRIMARY KEY,
   ORDERDATE datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
   CUSTOMER_ID INTEGER NOT NULL,
   SPECIAL_NOTES VARCHAR(255),
   FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID)
 );
 CREATE TABLE SALES(
   ORDER_ID INTEGER,
   DONUT_ID INTEGER,
   QUANTITY INTEGER NOT NULL,
   PRIMARY KEY (ORDER_ID, DONUT_ID),
   FOREIGN KEY (DONUT_ID) REFERENCES DONUTS (DONUT_ID),
   FOREIGN KEY(ORDER_ID) REFERENCES ORDERS (ORDER_ID)
 );
```

Below is all the insert statements that populate the tables within the database.

```
INSERT INTO CUSTOMER(CUSTOMER_ID, FNAME, LNAME, HPHONE)
VALUES(1, "Walter", "Smith", 809-603-9283);

INSERT INTO MAIL_ADDRESS(CUSTOMER_ID, STREET, APT, CITY, STATE, ZIP)
VALUES(1, "1250 Newton", 35, "Seattle", "WA", 98102);

INSERT INTO DONUTS(DONUT_ID, DNAME, DONUT_DESCRIPT, DONUT_PRICE)
VALUES(1, "Plane", "Plain Donut", 1.50),
(2, "Glazed", "Glazed Donut", 1.75),
(3, "Cinnamon", "Cinnamon Donut", 1.75),
(4, "Chocolate", "Chocolate Donut", 1.75),
(5, "Sprinkle", "Sprinkle Donut", 1.75),
(6, "Gluten-Free", "Gluten-free Donut", 2.00);

INSERT INTO ORDERS(ORDER_ID, CUSTOMER_ID, SPECIAL_NOTES)
VALUE(1, 1, "Please include plates and napkins.");
INSERT INTO SALES(ORDER_ID, DONUT_ID, QUANTITY)
VALUES(1, 1,1),
(1, 2, 5),
(1, 3, 12),
(1, 4, 3),
(1, 5, 4),
(1, 6, 5);
```

An index is created on invoice to assist with data searching on donut name.

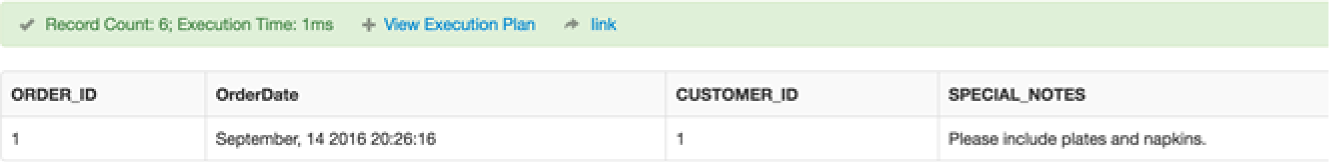
```
create index donutInfo on donuts(dname);
```

The view was created to display all data that was in the sales order invoice.

```
CREATE VIEW INVOICE AS
SELECT CUSTOMER.CUSTOMER_ID, CONCAT(CUSTOMER.FNAME,"
",CUSTOMER.LNAME) AS COSTOMER_NAME,
CUSTOMER.HPHONE, MAIL_ADDRESS.STREET, MAIL_ADDRESS.APT,
MAIL_ADDRESS.CITY, MAIL_ADDRESS.ZIP,
ORDERS.ORDER_ID, SALES.DONUT_ID, DONUTS.DNAME, DONUTS.DONUT_DISCRIPT,
DONUTS.DONUT_PRICE
FROM CUSTOMER JOIN MAIL_ADDRESS ON CUSTOMER.CUSTOMER_ID =
MAIL_ADDRESS.CUSTOMER_ID
JOIN ORDERS ON MAIL_ADDRESS.CUSTOMER_ID = ORDERS.CUSTOMER_ID
JOIN SALES ON ORDERS.ORDER_ID = SALES.ORDER_ID
RIGHT JOIN DONUTS ON SALES.DONUT_ID = DONUTS.DONUT_ID
```

Hear is a query that displays the special_notes and customer.

```
SELECT SPECIAL_NOTES, CUSTOMER_ID FROM ORDERS WHERE ORDER_ID = 1;
```

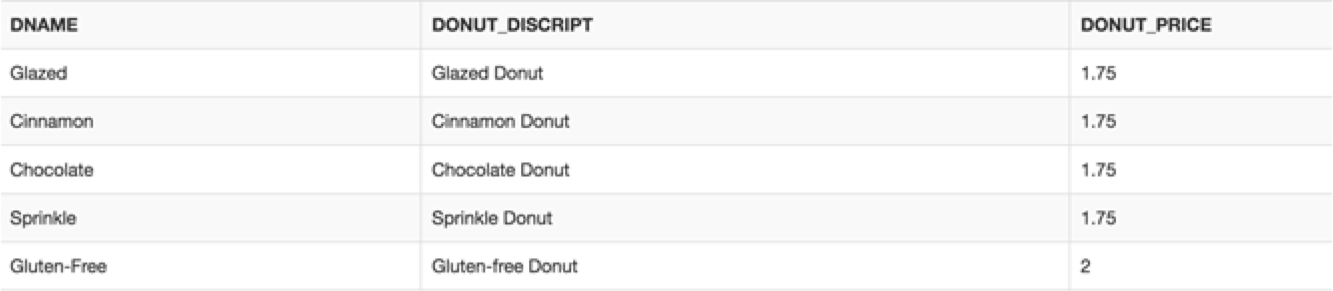


✓ Record Count: 6; Execution Time: 1ms

ORDER_ID	OrderDate	CUSTOMER_ID	SPECIAL_NOTES
1	September, 14 2016 20:26:16	1	Please include plates and napkins.

This is a simple select where query.

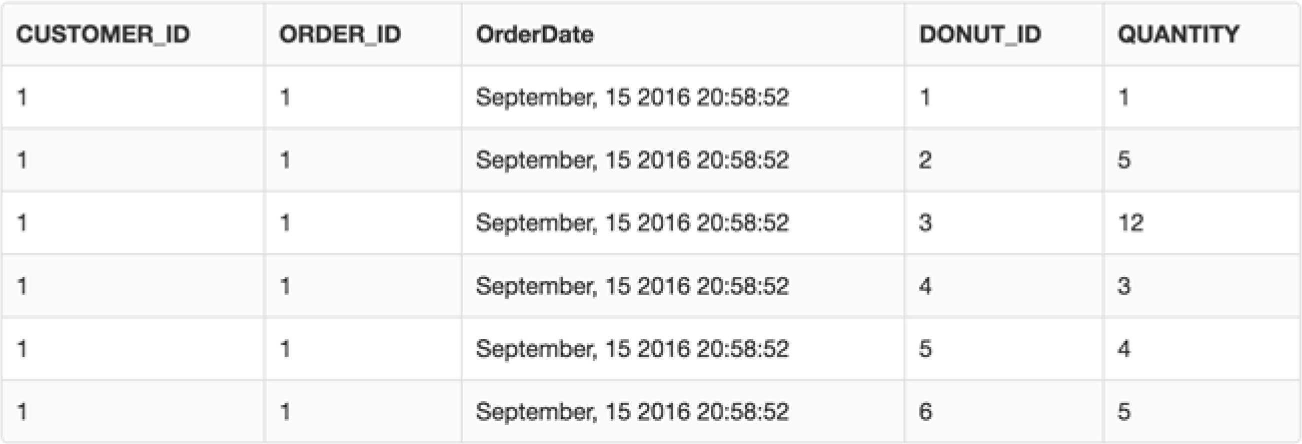
```
SELECT DNAME, DONUT_DISCRIPT, DONUT_PRICE FROM DONUTS WHERE
DONUT_PRICE >= 1.75
```



DNAME	DONUT_DISCRIPT	DONUT_PRICE
Glazed	Glazed Donut	1.75
Cinnamon	Cinnamon Donut	1.75
Chocolate	Chocolate Donut	1.75
Sprinkle	Sprinkle Donut	1.75
Gluten-Free	Gluten-free Donut	2

Hear is a complex join query.

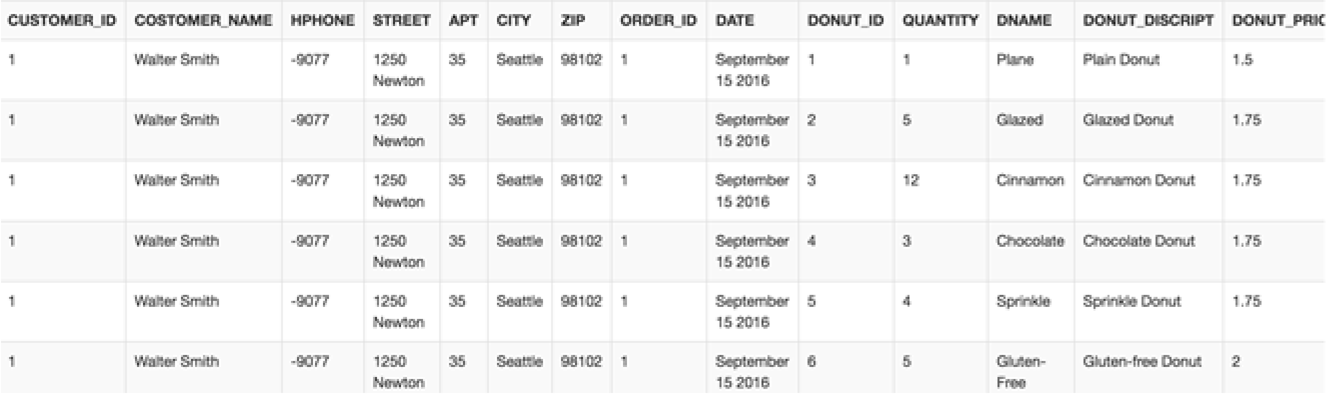
```
SELECT CUSTOMER.CUSTOMER_ID, ORDERS.ORDER_ID, ORDERS.ORDERDATE,
SALES.DONUT_ID, QUANTITY FROM SALES
JOIN ORDERS ON SALES.ORDER_ID = ORDERS.ORDER_ID JOIN CUSTOMER ON
ORDERS.CUSTOMER_ID=CUSTOMER.CUSTOMER_ID;
```



CUSTOMER_ID	ORDER_ID	OrderDate	DONUT_ID	QUANTITY
1	1	September, 15 2016 20:58:52	1	1
1	1	September, 15 2016 20:58:52	2	5
1	1	September, 15 2016 20:58:52	3	12
1	1	September, 15 2016 20:58:52	4	3
1	1	September, 15 2016 20:58:52	5	4
1	1	September, 15 2016 20:58:52	6	5

This is all data that is contained the view invoices complied from all four tables in a complex join

```
SELECT * FROM INVOICES;
```



CUSTOMER_ID	COSTOMER_NAME	HPHONE	STREET	APT	CITY	ZIP	ORDER_ID	DATE	DONUT_ID	QUANTITY	DNAME	DONUT_DISCRIPT	DONUT_PRIC
1	Walter Smith	-9077	1250 Newton	35	Seattle	98102	1	September 15 2016	1	1	Plane	Plain Donut	1.5
1	Walter Smith	-9077	1250 Newton	35	Seattle	98102	1	September 15 2016	2	5	Glazed	Glazed Donut	1.75
1	Walter Smith	-9077	1250 Newton	35	Seattle	98102	1	September 15 2016	3	12	Cinnamon	Cinnamon Donut	1.75
1	Walter Smith	-9077	1250 Newton	35	Seattle	98102	1	September 15 2016	4	3	Chocolate	Chocolate Donut	1.75
1	Walter Smith	-9077	1250 Newton	35	Seattle	98102	1	September 15 2016	5	4	Sprinkle	Sprinkle Donut	1.75
1	Walter Smith	-9077	1250 Newton	35	Seattle	98102	1	September 15 2016	6	5	Gluten-Free	Gluten-free Donut	2