

Section One – Relating Data

1. *Creating the Table Structure* – Create the Phone and Customer tables, including all their columns, datatypes, and constraints, and the foreign key constraint.

```
1  -- Step One
2  CREATE TABLE Phone(
3      phone_id DECIMAL(12) NOT NULL PRIMARY KEY,
4      phone_model VARCHAR(32) NOT NULL,
5      phone_price DECIMAL(6,2) NOT NULL,
6      release_date DATE NOT NULL);
7
8  CREATE TABLE Customer(
9      customer_id DECIMAL(12) NOT NULL PRIMARY KEY,
10     customer_email VARCHAR(64) NOT NULL,
11     customer_name VARCHAR(64) NOT NULL,
12     phone_id DECIMAL(12) NULL);
13
14 ALTER TABLE Customer
15 ADD CONSTRAINT customer_fk
16 FOREIGN KEY (phone_id)
17 REFERENCES Phone(phone_id);
18 |
```

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 52 msec.

2. *Populating the Tables* – Insert the following rows into the Phone table.

Phone 1

name = Apple iPhone X
release_date = 11/03/2017
price = \$379

Phone 2

name = Galaxy S21+
release_date = 01/29/2021
price = \$799

Phone 3

name = Xenos 360
release_date = 03/22/2021
price = \$1,024

Phone 4

name = Meridian Duplex

release_date = 05/15/2021
price = \$462

Insert five customers of your choosing into the Customer table. Please note the following. *The first customer must not be associated with any phone because they have not yet purchased any phone, and the first phone (Apple iPhone X) must not be associated with any customer because no one has yet purchased it.* The rest of the customers should be associated with phones, and the rest of the phones should be associated with customers.

Select all rows in both tables to view what you inserted.

```
19 -- Step Two
20 INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
21 VALUES
22     (1, 'Apple iPhone X', 379.00, '2017-11-03'),
23     (2, 'Galaxy S21+', 799.00, '2021-01-29'),
24     (3, 'Xenos 360', 1024.00, '2021-03-22'),
25     (4, 'Meridian Duplex', 462.00, '2021-05-15');
26
27 INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
28 VALUES
29     (1, 'katherine@gmail.com', 'Katherine', NULL),
30     (2, 'maddie@gmail.com', 'Maddie', 2),
31     (3, 'laura@gmail.com', 'Laura', 3),
32     (4, 'collin@gmail.com', 'Collin', 2),
33     (5, 'josie@gmail.com', 'Josie', 4);
34
35 SELECT *
36 FROM Phone;
37
38 SELECT *
39 FROM Customer;
```

Data Output Messages Notifications

	phone_id [PK] numeric (12)	phone_model character varying (32)	phone_price numeric (6,2)	release_date date
1	1	Apple iPhone X	379.00	2017-11-03
2	2	Galaxy S21+	799.00	2021-01-29
3	3	Xenos 360	1024.00	2021-03-22
4	4	Meridian Duplex	462.00	2021-05-15

	customer_id [PK] numeric (12)	customer_email character varying (64)	customer_name character varying (64)	phone_id numeric (12)
1	1	katherine@gmail.com	Katherine	[null]
2	2	maddie@gmail.com	Maddie	2
3	3	laura@gmail.com	Laura	3
4	4	collin@gmail.com	Collin	2
5	5	josie@gmail.com	Josie	4

3. *Invalid Reference Attempt* – As an exercise, attempt to insert a Customer that references a Phone that doesn't exist. Summarize:
- why the insertion failed, and
 - how you would interpret the error message from your RDBMS so that you know that the error indicates the Phone reference is invalid.

```

41 -- Step 3
42 INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
43 VALUES (6, 'amy@gmail.com', 'Amy', 5);
44

```

Data Output [Messages](#) Notifications

ERROR: Key (phone_id)=(5) is not present in table "phone".insert or update on table "customer" violates foreign key constraint "customer_fk"

ERROR: insert or update on table "customer" violates foreign key constraint "customer_fk"
SQL state: 23503

Detail: Key (phone_id)=(5) is not present in table "phone".

The insertion failed because the phone_id does not exist in the Phone table. For the row to be added it has to reference a row that exists in Phone using the phone_id foreign key.

The error message lets us know that the value 5 for phone_id is invalid because it's not in the Phone table. It says that the insert we tried to do on Customer violates the foreign key.

4. *Listing Matches* – With a single SQL query, fulfill the following request:

List the names of the Phones that have Customers, and the names of all the Customers that have a Phone.

```

45 -- Step 4
46 SELECT phone_model
47 FROM Phone
48 JOIN Customer ON Phone.phone_id = Customer.phone_id;
49

```

Data Output Messages Notifications

	phone_model character varying (32) 🔒
1	Galaxy S21+
2	Xenos 360
3	Galaxy S21+
4	Meridian Duplex

```

50 SELECT customer_name
51 FROM Customer
52 JOIN Phone ON Customer.phone_id = Phone.phone_id;
53

```

Data Output Messages Notifications

	customer_name character varying (64) 🔒
1	Maddie
2	Laura
3	Collin
4	Josie

From a technical SQL perspective, explain why some rows in the Phone table and some rows in the Customers table were not listed.

The reason some rows are taken out is because those values are null in the whole complete table so they don't get selected when we select only phone_model or customer_name. For example, Katherine doesn't own a phone so when we join Phone with Customer the phone_id is null and so it creates a row of nulls thus resulting in a null model_id.

5. *Listing All from One Table* – Fulfill the following request:

List the names and release date of all Phones whether or not they have been purchased by Customers. For the Phones that were purchased by customers Customers, list the names of the Customers that have those Phones. Order the list by the release date, oldest to newest.

There are two kinds of joins that can be used to satisfy this request. Write two queries using each type of join to satisfy this request.

```
54 -- Step 5
55 SELECT phone_model, release_date
56 FROM Phone;
57
58 SELECT customer_name
59 FROM Customer
60 JOIN Phone ON Customer.phone_id = Phone.phone_id
61 ORDER BY release_date ASC;
62
63
```

Data Output		Messages	Notifications
	customer_name character varying (64) 🔒		
1	Maddie		
2	Collin		
3	Laura		
4	Josie		

```
63 SELECT customer_name, release_date
64 FROM Customer
65 INNER JOIN Phone ON Customer.phone_id = Phone.phone_id
66 ORDER BY release_date ASC;
67
68
```

Data Output			Messages	Notifications
	customer_name character varying (64) 🔒	release_date date 🔒		
1	Maddie	2021-01-29		
2	Collin	2021-01-29		
3	Laura	2021-03-22		
4	Josie	2021-05-15		

6. Listing All from Another Table – Fulfill the following request:

List the names and emails of all Customers whether or not they have purchased a Phone,

and the names of the Phones which Customers have purchased. Order the list by Customer email in reverse alphabetical order.

Just as with step #5, there are two kinds of joins that can be used to satisfy this request. Write two queries using each type of join to satisfy this request.

```
68 -- Step 6
69 SELECT customer_name, customer_email
70 FROM Customer;
71
72 SELECT customer_name, customer_email, phone_model
73 FROM Phone
74 JOIN Customer ON Phone.phone_id = Customer.phone_id
75 ORDER BY customer_email DESC;
```

Data Output Messages Notifications

	customer_name character varying (64)	customer_email character varying (64)	phone_model character varying (32)
1	Maddie	maddie@gmail.com	Galaxy S21+
2	Laura	laura@gmail.com	Xenos 360
3	Josie	josie@gmail.com	Meridian Duplex
4	Collin	collin@gmail.com	Galaxy S21+

```
77 SELECT customer_name, customer_email, phone_model
78 FROM Phone
79 INNER JOIN Customer ON Phone.phone_id = Customer.phone_id
80 ORDER BY customer_email DESC;
81
```

Data Output Messages Notifications

	customer_name character varying (64)	customer_email character varying (64)	phone_model character varying (32)
1	Maddie	maddie@gmail.com	Galaxy S21+
2	Laura	laura@gmail.com	Xenos 360
3	Josie	josie@gmail.com	Meridian Duplex
4	Collin	collin@gmail.com	Galaxy S21+

7. Listing All from Both Tables – Fulfill the following request with a single SQL query:

List the names of all Phones and the emails of all Customers, as well as which Phones have which Customers. Order the list alphabetically by Phone name then by Customer name.

```

82 -- Step 7
83 SELECT *
84 FROM Phone
85 JOIN Customer ON Phone.phone_id = Customer.phone_id
86 ORDER BY phone_model, customer_name;
87

```

Data Output Messages Notifications								
	phone_id numeric (12)	phone_model character varying (32)	phone_price numeric (6,2)	release_date date	customer_id numeric (12)	customer_email character varying (64)	customer_name character varying (64)	phone_id numeric (12)
1	2	Galaxy S21+	799.00	2021-01-29	4	collin@gmail.com	Collin	2
2	2	Galaxy S21+	799.00	2021-01-29	2	maddie@gmail.com	Maddie	2
3	4	Meridian Duplex	462.00	2021-05-15	5	josie@gmail.com	Josie	4
4	3	Xenos 360	1024.00	2021-03-22	3	laura@gmail.com	Laura	3

Section Two – Expressing Data

8. Formatting as Money – Fulfill the following request with a single query:

The managers of the Phone store want to review their prices. List the names and prices of all Phones, making sure to format the price monetarily in U.S. dollars (for example, “\$379.00”).

```

88 -- Step 8
89 SELECT phone_model, to_char(phone_price, '$9999.00')
90 FROM Phone;
91
92

```

Data Output Messages Notifications		
	phone_model character varying (32)	to_char text
1	Apple iPhone X	\$ 379.00
2	Galaxy S21+	\$ 799.00
3	Xenos 360	\$ 1024.00
4	Meridian Duplex	\$ 462.00

9. Using Expressions – Fulfill the following request with a single query:

The managers of the Phone store are looking to increase purchases of Phones by lowering prices by \$50. List the names and discounted prices of all Phones, making sure to format the price monetarily in U.S. dollars.

```

92 -- Step 9
93 SELECT phone_model, to_char(phone_price - 50, '$9999.00')
94 FROM Phone;
95
96

```

Data Output Messages Notifications		
	phone_model character varying (32)	to_char text
1	Apple iPhone X	\$ 329.00
2	Galaxy S21+	\$ 749.00
3	Xenos 360	\$ 974.00
4	Meridian Duplex	\$ 412.00

10. *Advanced Formatting* – Fulfill the following request with a single query:

The managers want to determine what Phone each Customer has purchased as well as its price, and wants the list ordered by Customer name. For example, if the Apple iPhone X has a price of \$379, and has two Customers – John Doe and Jane Doe – the results would have two lines for this Phone:

John Doe (Apple iPhone X - \$379.00)

Jane Doe (Apple iPhone X - \$379.00)

```

96 -- Step 10
97 SELECT customer_name || ' (' || phone_model || ' - ' || to_char(phone_price, '$9999.00') || ')'
98 FROM Phone
99 JOIN Customer ON Phone.phone_id = Customer.phone_id;
100

```

Data Output Messages Notifications		
	?column? text	
1	Maddie (Galaxy S21+ - \$ 799.00)	
2	Laura (Xenos 360 - \$ 1024.00)	
3	Collin (Galaxy S21+ - \$ 799.00)	
4	Josie (Meridian Duplex - \$ 462.00)	

Section Three – Advanced Data Expression

11. *Evaluating Boolean Expressions* – Indicate the final values for each of the Boolean expressions below. You must show your work for full credit, by showing the value of each operation step-by-step.

a. (true AND false) OR (false AND true)

False OR False

False

b. (true OR true) AND NOT(false OR true) AND (true AND true)

True AND NOT True AND True

True AND False AND True

False AND True

False

c. NOT((false OR false) AND NOT(true AND true) AND (true OR false))

NOT(False AND NOT True AND True)

NOT(False AND False AND True)

NOT(False AND True)

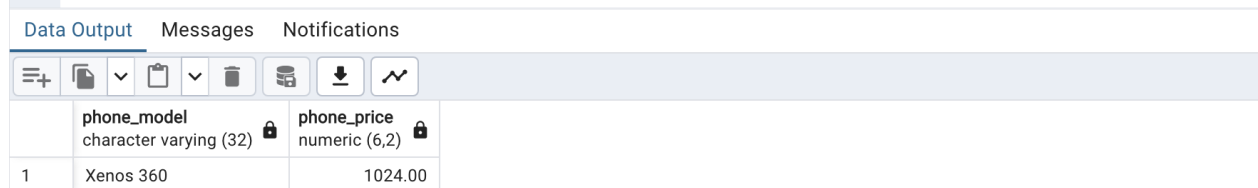
NOT(False)

True

12. *Using Boolean Expressions in Queries* – Address the following scenarios.

a. Any Phone matching the following condition is considered a high-end Phone for the store: Any Phone, except for the “Apple iPhone X” Phone, that is available on or after 05/01/2020, with a price of \$900.00 or higher, is a high-end Phone. Write a query that shows the name and price of all high-end Phones. It’s fine if you’d like to insert another row of Phones to become the high-end Phone.

```
101 -- Step 11
102 SELECT phone_model, phone_price
103 FROM Phone
104 WHERE phone_model != 'Apple iPhone X' AND release_date >= '2020-05-01' AND phone_price >= 900.00;
105
```



	phone_model character varying (32)	phone_price numeric (6,2)
1	Xenos 360	1024.00

b. The management company also has one *deluxe Phone* that sets it apart from other Phones. First, define your own conditions for this Phone, making sure the conditions include the Phone name, release date, and the phone price. Then write a query that shows the name and price of the Phone. It’s fine if you’d like to insert another row of Phones to become the deluxe Phone.

Any phone that is available after 03/05/2021, with a price of more than \$1000, and not called the Xenos 360.

```

106 INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
107 VALUES (5, 'Apple iPhone 13', 1200.00, '2021-11-03');
108
109 SELECT phone_model, phone_price
110 FROM Phone
111 WHERE phone_model != 'Xenos 360' AND release_date > '2021-03-05' AND phone_price > 1000.00;
112

```

Data Output	Messages	Notifications
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>		
phone_model	phone_price	
character varying (32)	numeric (6,2)	
1	Apple iPhone 13	1200.00

13. Using Generated Columns – Address the following.

- Define a new generated column named *reduced_price*, which gives a lower price for the Phone for when the store wants to increase purchases by lowering prices (such as after the Christmas holiday shopping season). You determine the percentage or fixed value discount for these Phones. Then write a query that lists out the name of all Phones, along with their regular and reduced prices.

```

113 -- Step 13
114 ALTER TABLE Phone
115 ADD COLUMN reduced_price DECIMAL(6,2) GENERATED ALWAYS AS (phone_price * 0.75) STORED;
116
117 SELECT phone_model, phone_price, reduced_price
118 FROM Phone;
119

```

Data Output	Messages	Notifications
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>		
phone_model	phone_price	reduced_price
character varying (32)	numeric (6,2)	numeric (6,2)
1	Apple iPhone X	379.00
2	Galaxy S21+	799.00
3	Xenos 360	1024.00
4	Meridian Duplex	462.00
5	Apple iPhone 13	1200.00

- Address #12a again in a different way. First, define a generated column named *is_high_end* on the Phone table, which indicates whether it's a high-end Phone or not. Then write a query that lists only those Phones. Include relevant columns in the result.

```

120 ALTER TABLE Phone
121 ADD COLUMN is_high_end BOOLEAN GENERATED ALWAYS AS (phone_price > 700) STORED;
122
123 SELECT phone_model, phone_price, reduced_price
124 FROM Phone
125 WHERE is_high_end = TRUE;

```

Data Output Messages Notifications

	phone_model character varying (32) 🔒	phone_price numeric (6,2) 🔒	reduced_price numeric (6,2) 🔒
1	Galaxy S21+	799.00	599.25
2	Xenos 360	1024.00	768.00
3	Apple iPhone 13	1200.00	900.00