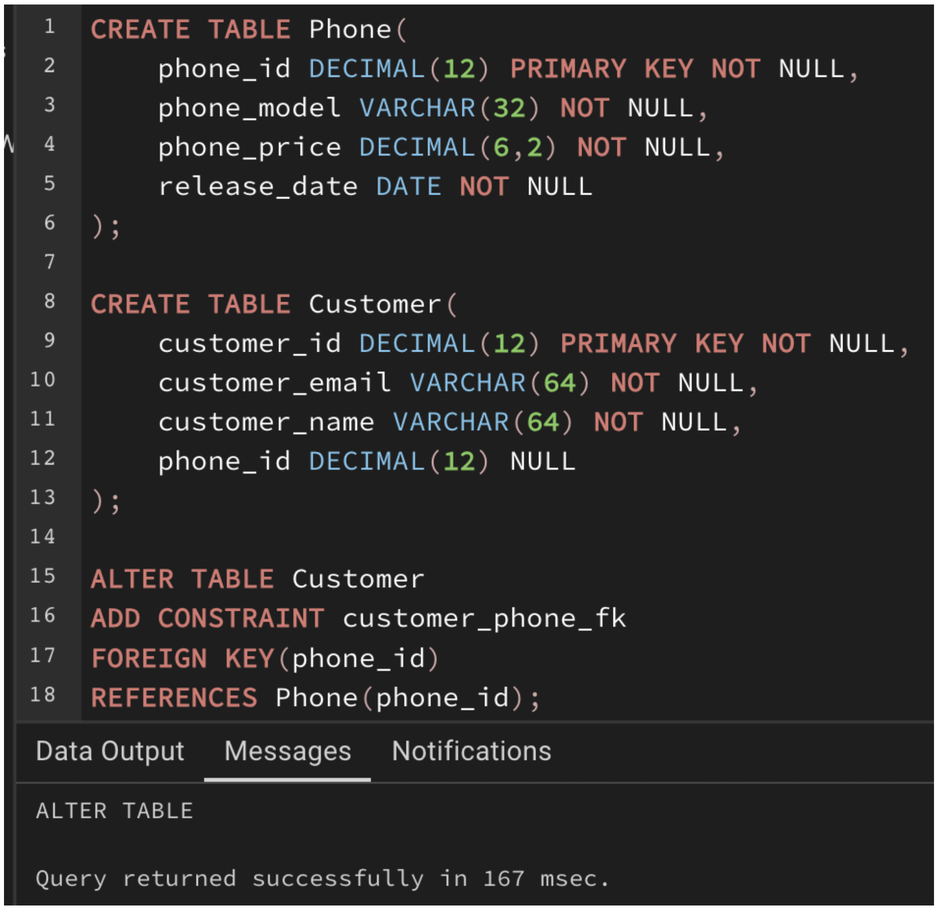
Section One – Relating Data

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



```
CREATE TABLE Phone(
    phone_id DECIMAL(12) PRIMARY KEY NOT NULL,
    phone_model VARCHAR(32) NOT NULL,
    phone_price DECIMAL(6,2) NOT NULL,
    release_date DATE NOT NULL
);

CREATE TABLE Customer(
    customer_id DECIMAL(12) PRIMARY KEY NOT NULL,
    customer_email VARCHAR(64) NOT NULL,
    customer_name VARCHAR(64) NOT NULL,
    phone_id DECIMAL(12) NULL
);

ALTER TABLE Customer
ADD CONSTRAINT customer_phone_fk
FOREIGN KEY(phone_id)
REFERENCES Phone(phone_id);
```

```
ALTER TABLE

Query returned successfully in 167 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

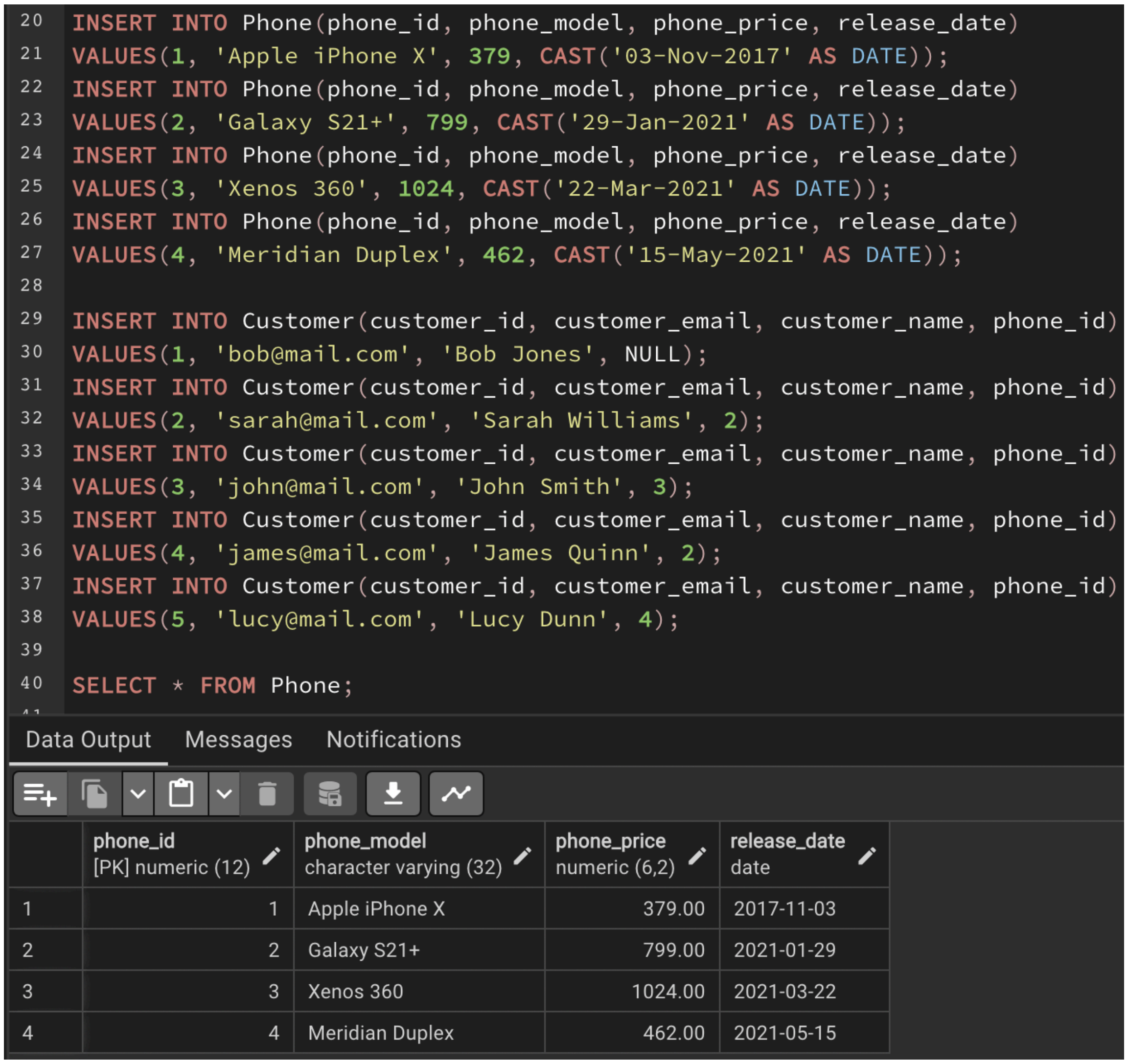
Phone 3
name = Xenos 360
release_date = 03/22/2021

price = $1,024

Phone 4
name = Meridian Duplex
release_date = 05/15/2021
price = $462

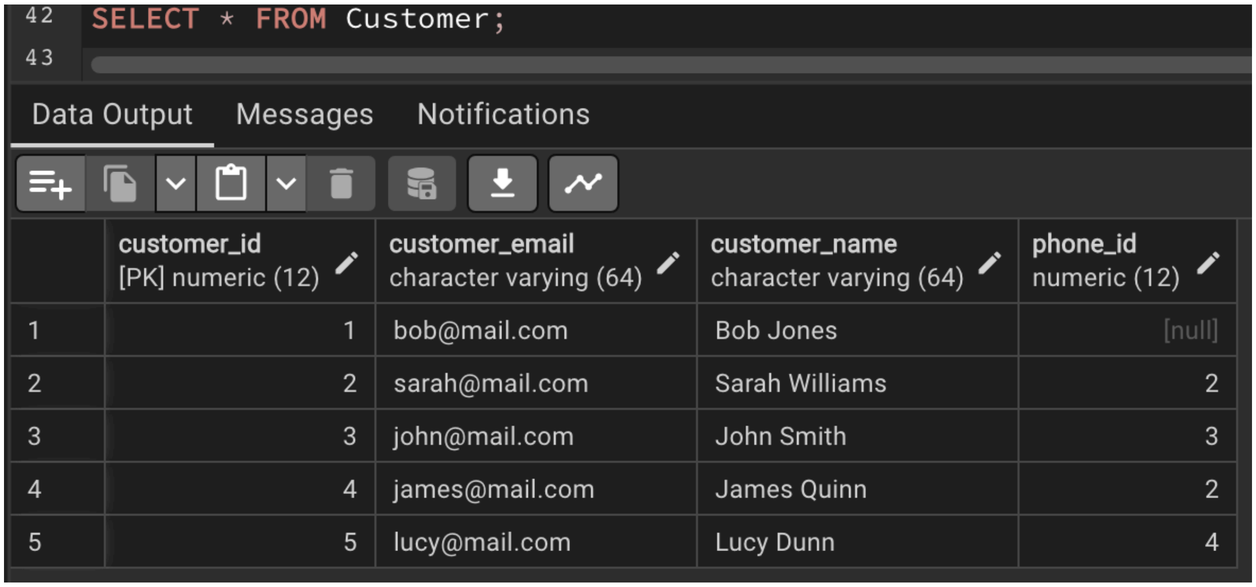
Insert five customers of your choosing into the Customer table. Please note the following. *The first customer must not be associated with any phone because they have not yet purchased any phone, and the first phone (Apple iPhone X) must not be associated with any customer because no one has yet purchased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



```
INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
VALUES(1, 'Apple iPhone X', 379, CAST('03-Nov-2017' AS DATE));
INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
VALUES(2, 'Galaxy S21+', 799, CAST('29-Jan-2021' AS DATE));
INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
VALUES(3, 'Xenos 360', 1024, CAST('22-Mar-2021' AS DATE));
INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
VALUES(4, 'Meridian Duplex', 462, CAST('15-May-2021' AS DATE));

INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
VALUES(1, 'bob@mail.com', 'Bob Jones', NULL);
INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
VALUES(2, 'sarah@mail.com', 'Sarah Williams', 2);
INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
VALUES(3, 'john@mail.com', 'John Smith', 3);
INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
VALUES(4, 'james@mail.com', 'James Quinn', 2);
INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
VALUES(5, 'lucy@mail.com', 'Lucy Dunn', 4);

SELECT * FROM Phone;
```

	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

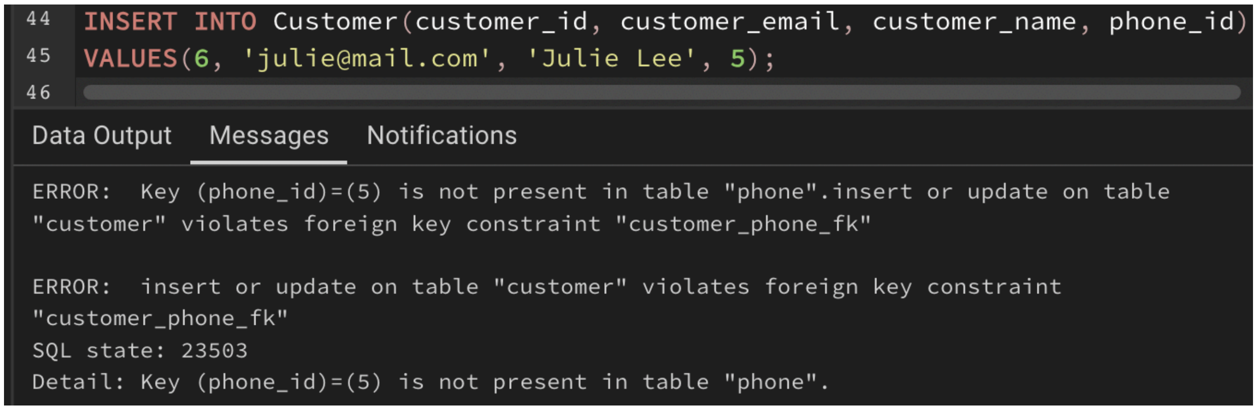


```
SELECT * FROM Customer;
```

Data Output   Messages   Notifications

	customer_id [PK] numeric (12)	customer_email character varying (64)	customer_name character varying (64)	phone_id numeric (12)
1	1	bob@mail.com	Bob Jones	[null]
2	2	sarah@mail.com	Sarah Williams	2
3	3	john@mail.com	John Smith	3
4	4	james@mail.com	James Quinn	2
5	5	lucy@mail.com	Lucy Dunn	4

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



```
INSERT INTO Customer(customer_id, customer_email, customer_name, phone_id)
VALUES(6, 'julie@mail.com', 'Julie Lee', 5);
```

Data Output   Messages   Notifications

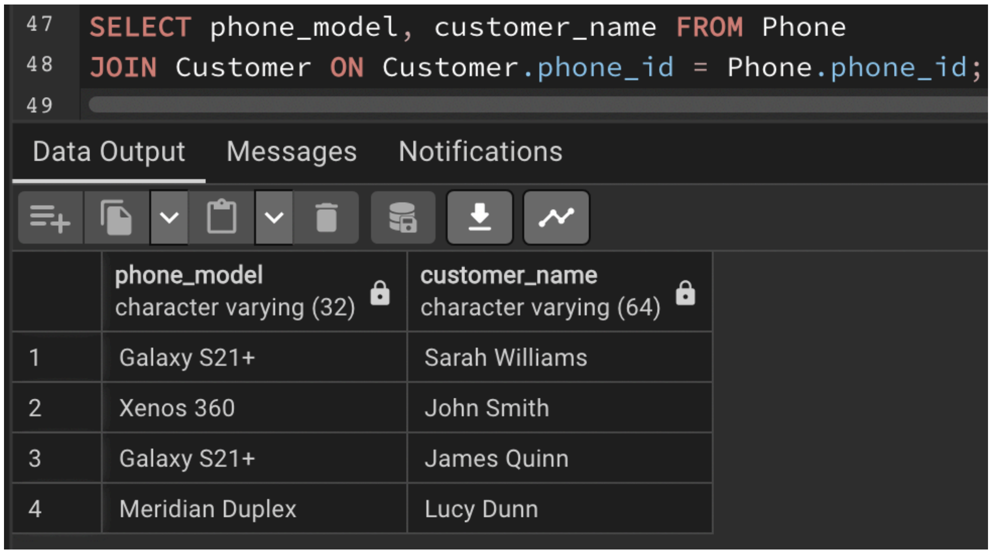
```
ERROR:  Key (phone_id)=(5) is not present in table "phone".insert or update on table
"customer" violates foreign key constraint "customer_phone_fk"

ERROR:  insert or update on table "customer" violates foreign key constraint
"customer_phone_fk"
SQL state: 23503
Detail: Key (phone_id)=(5) is not present in table "phone".
```

The insertion failed because it violates the foreign key constraint as it attempts to insert a phone_id that does not already exist in the Phone table into the Customer table. The error message is straightforward in that it tells us that the key with value 5 is not in the Phone table and tells us the name of the foreign key constraint to let us know that our attempt to insert is invalid.

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.

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



```
SELECT phone_model, customer_name FROM Phone
JOIN Customer ON Customer.phone_id = Phone.phone_id;
```

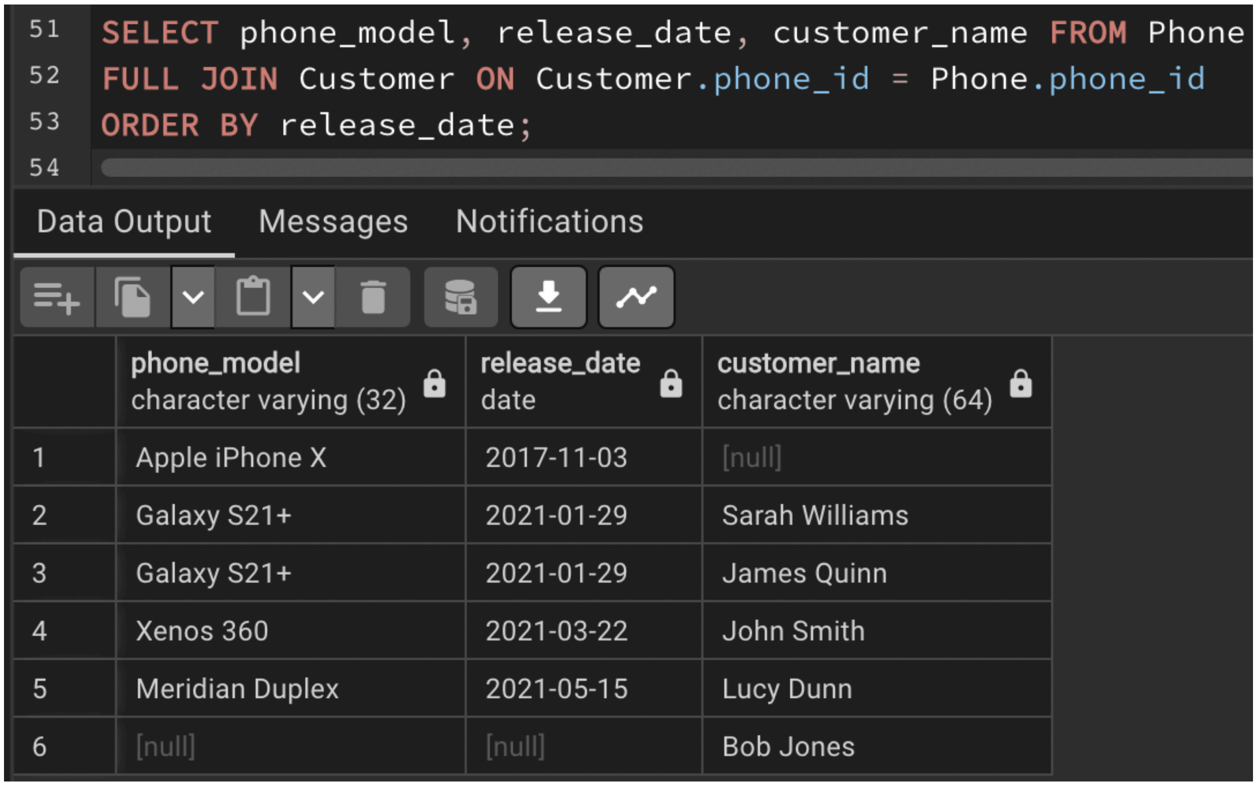
	phone_model character varying (32)	customer_name character varying (64)
1	Galaxy S21+	Sarah Williams
2	Xenos 360	John Smith
3	Galaxy S21+	James Quinn
4	Meridian Duplex	Lucy Dunn

Some rows in both the Phone and Customer table were not included in the list because they do not match the criteria created by the Boolean expression that results from using the ON keyword in our join query. Because this is an inner join, all rows that do not match the join condition are not included in the list.

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.

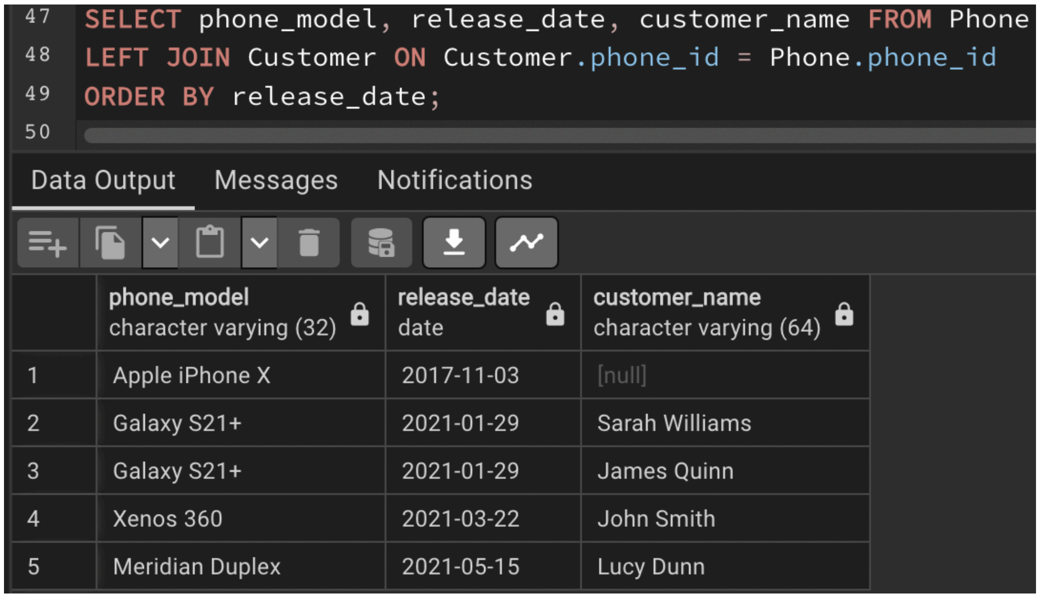
   There are two kinds of joins that can be used to satisfy this request. Write two queries using each type of join to satisfy this request.



```
SELECT phone_model, release_date, customer_name FROM Phone
FULL JOIN Customer ON Customer.phone_id = Phone.phone_id
ORDER BY release_date;
```

Data Output | Messages | Notifications

	phone_model character varying (32)	release_date date	customer_name character varying (64)
1	Apple iPhone X	2017-11-03	[null]
2	Galaxy S21+	2021-01-29	Sarah Williams
3	Galaxy S21+	2021-01-29	James Quinn
4	Xenos 360	2021-03-22	John Smith
5	Meridian Duplex	2021-05-15	Lucy Dunn
6	[null]	[null]	Bob Jones



```
SELECT phone_model, release_date, customer_name FROM Phone
LEFT JOIN Customer ON Customer.phone_id = Phone.phone_id
ORDER BY release_date;
```

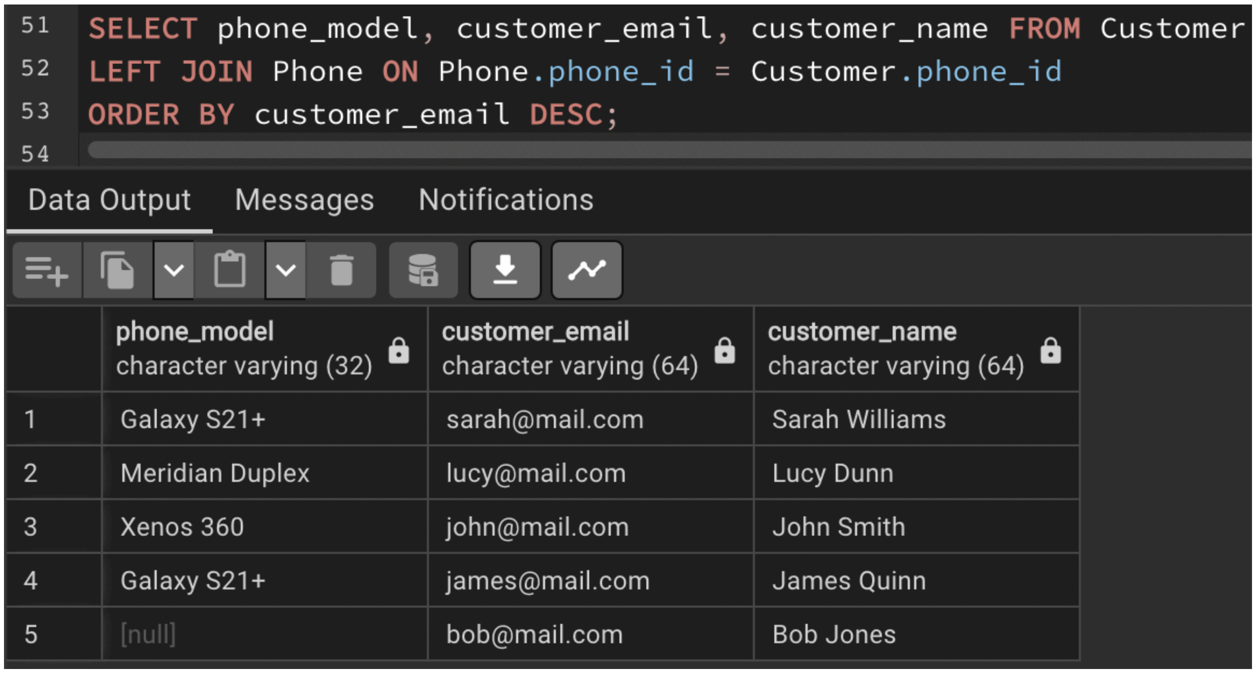
Data Output | Messages | Notifications

	phone_model character varying (32)	release_date date	customer_name character varying (64)
1	Apple iPhone X	2017-11-03	[null]
2	Galaxy S21+	2021-01-29	Sarah Williams
3	Galaxy S21+	2021-01-29	James Quinn
4	Xenos 360	2021-03-22	John Smith
5	Meridian Duplex	2021-05-15	Lucy Dunn

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

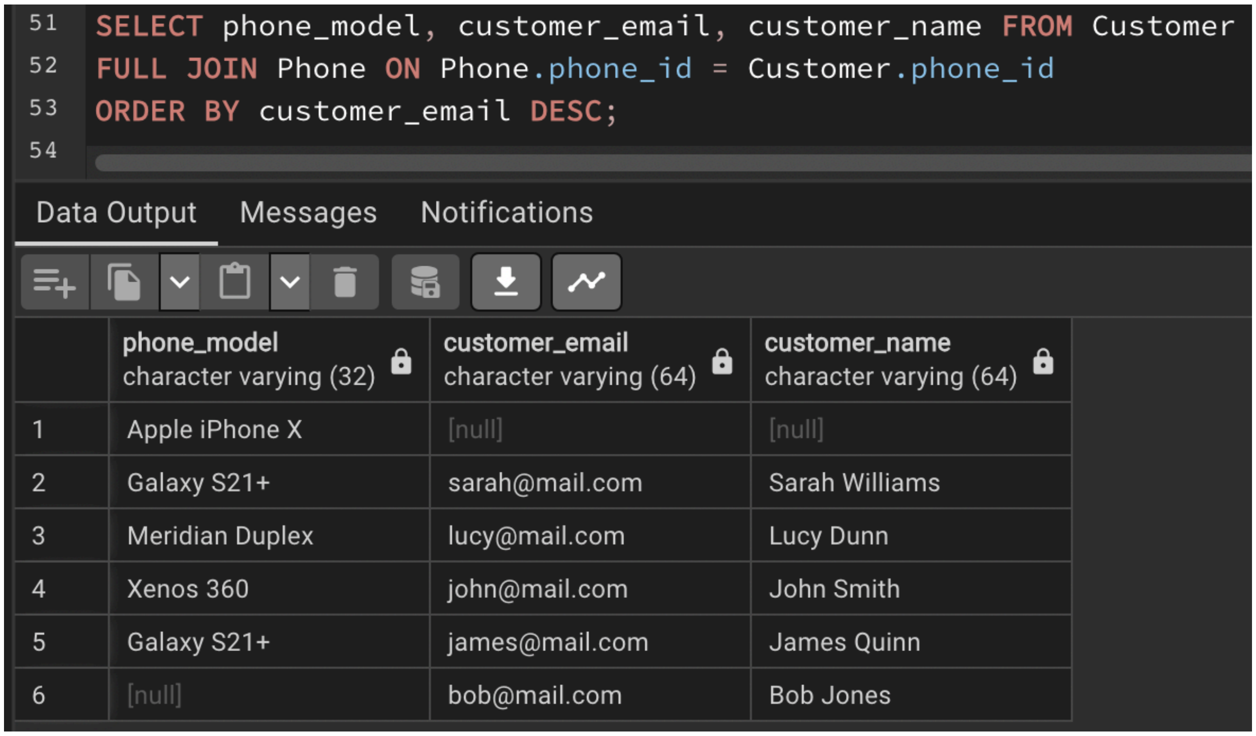
   List the names and emails of all Customers whether or not they have purchased a Phone, and the names of the Phones which Customers have purchased. Order the list by Customer email in reverse alphabetical order.

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.



```
SELECT phone_model, customer_email, customer_name FROM Customer
LEFT JOIN Phone ON Phone.phone_id = Customer.phone_id
ORDER BY customer_email DESC;
```

	phone_model character varying (32)	customer_email character varying (64)	customer_name character varying (64)
1	Galaxy S21+	sarah@mail.com	Sarah Williams
2	Meridian Duplex	lucy@mail.com	Lucy Dunn
3	Xenos 360	john@mail.com	John Smith
4	Galaxy S21+	james@mail.com	James Quinn
5	[null]	bob@mail.com	Bob Jones

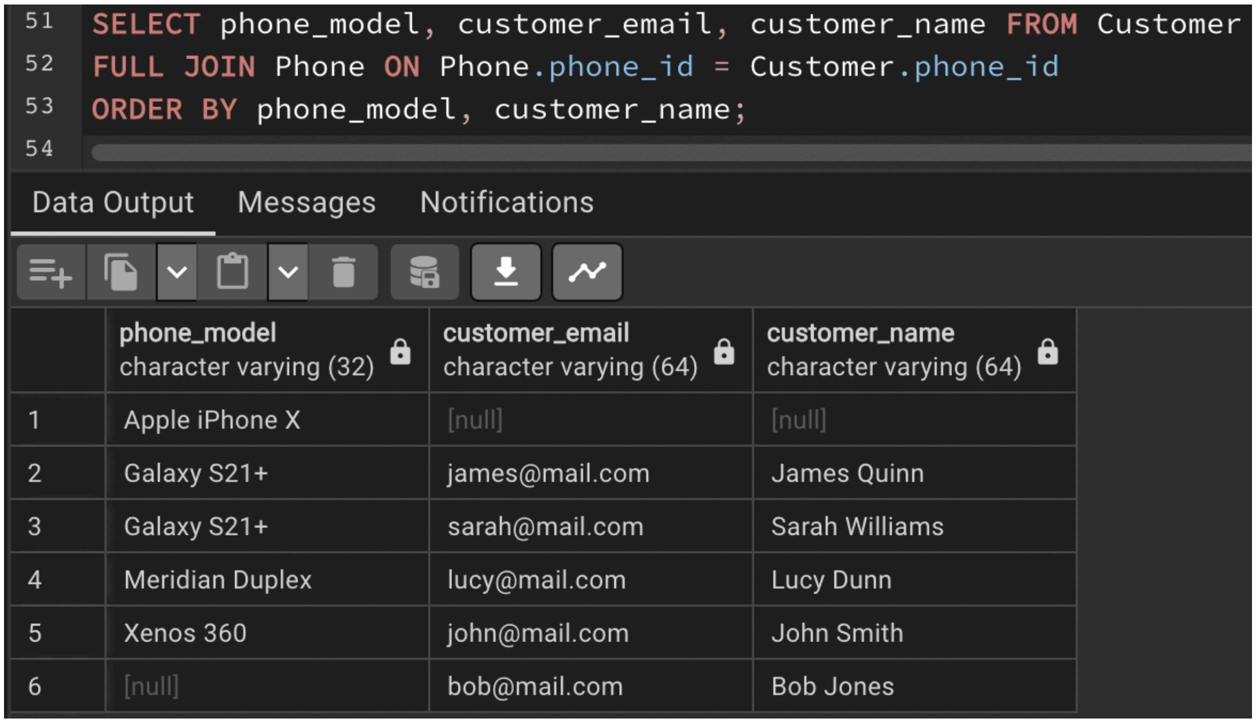


```
SELECT phone_model, customer_email, customer_name FROM Customer
FULL JOIN Phone ON Phone.phone_id = Customer.phone_id
ORDER BY customer_email DESC;
```

	phone_model character varying (32)	customer_email character varying (64)	customer_name character varying (64)
1	Apple iPhone X	[null]	[null]
2	Galaxy S21+	sarah@mail.com	Sarah Williams
3	Meridian Duplex	lucy@mail.com	Lucy Dunn
4	Xenos 360	john@mail.com	John Smith
5	Galaxy S21+	james@mail.com	James Quinn
6	[null]	bob@mail.com	Bob Jones

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.



```
SELECT phone_model, customer_email, customer_name FROM Customer
FULL JOIN Phone ON Phone.phone_id = Customer.phone_id
ORDER BY phone_model, customer_name;
```

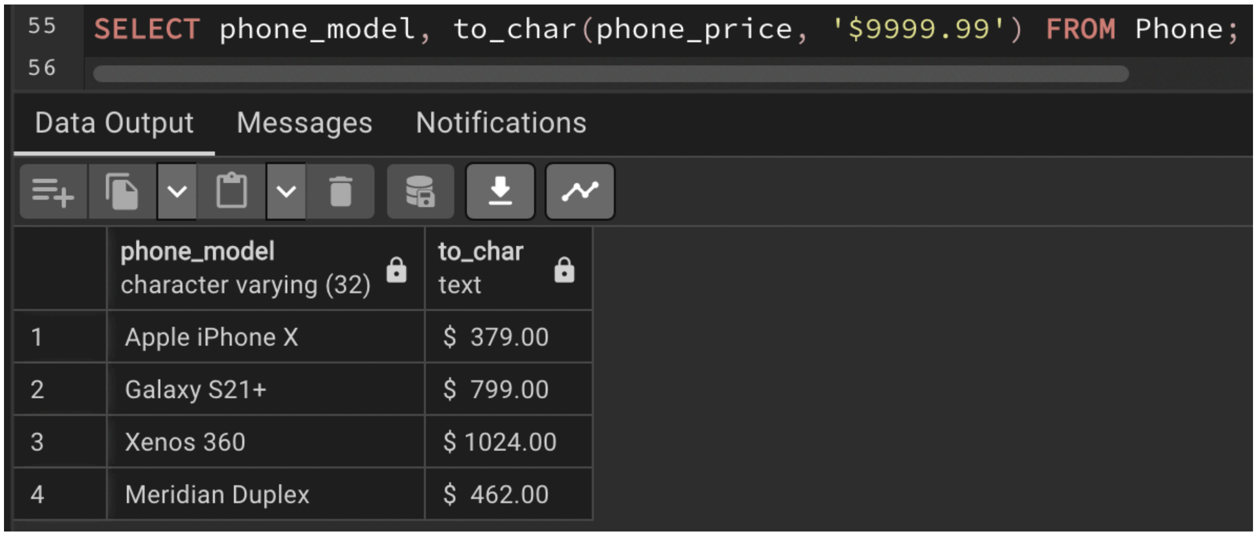
Data Output Messages Notifications

	phone_model character varying (32)	customer_email character varying (64)	customer_name character varying (64)
1	Apple iPhone X	[null]	[null]
2	Galaxy S21+	james@mail.com	James Quinn
3	Galaxy S21+	sarah@mail.com	Sarah Williams
4	Meridian Duplex	lucy@mail.com	Lucy Dunn
5	Xenos 360	john@mail.com	John Smith
6	[null]	bob@mail.com	Bob Jones

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").



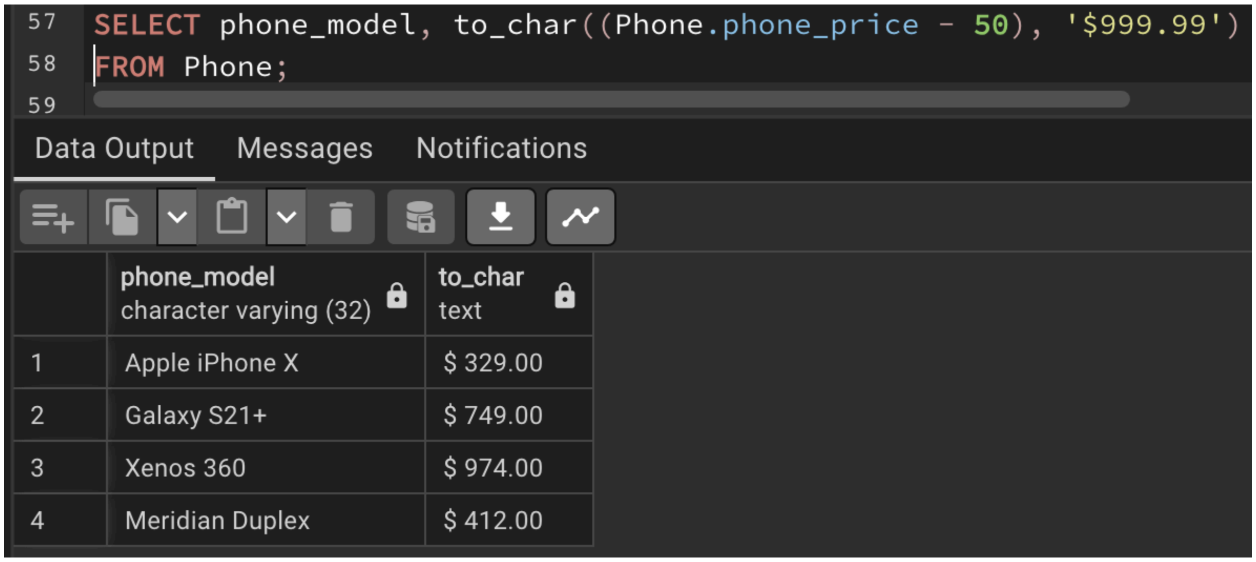
```
SELECT phone_model, to_char(phone_price, '$9999.99') FROM Phone;
```

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.



```sql
SELECT phone_model, to_char((Phone.phone_price - 50), '$999.99')
FROM Phone;
```

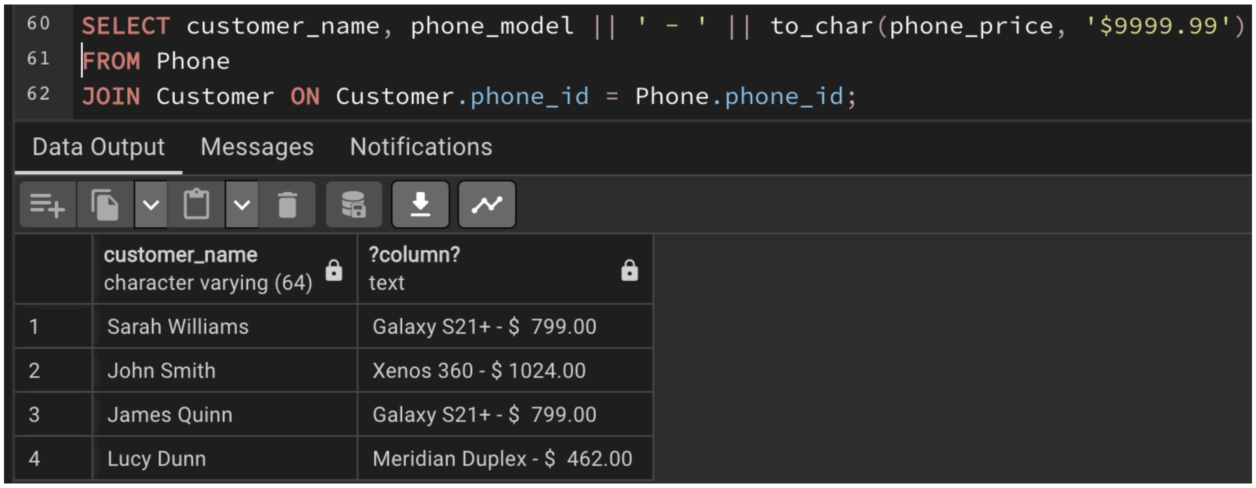
Data Output Messages Notifications

	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)



```sql
SELECT customer_name, phone_model || ' - ' || to_char(phone_price, '$9999.99')
FROM Phone
JOIN Customer ON Customer.phone_id = Phone.phone_id;
```

Data Output Messages Notifications

	customer_name character varying (64)	?column? text
1	Sarah Williams	Galaxy S21+ - $ 799.00
2	John Smith	Xenos 360 - $ 1024.00
3	James Quinn	Galaxy S21+ - $ 799.00
4	Lucy Dunn	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)

(True AND false) as well as (false AND true) both result in false because the result of an AND can only be true if both operands are true. Then, false OR false results in the final value being false, because an OR can only be false if both operands are false.

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

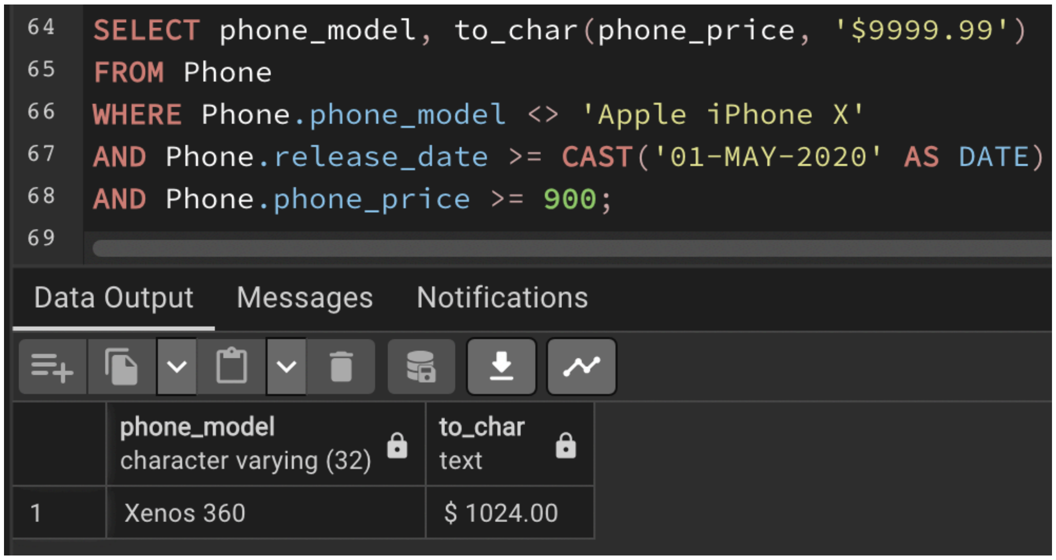
The first (true OR true) will be true because one operand is true, as well as the last (true AND true) because both operands are true. The (false OR true) in the middle will be true because one value is true, but the NOT preceding it will make it false because it switches the result. We end up with true AND false AND true, with the final result being false because an AND needs all operands to be true to result in a true.

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

(False OR false) will be true because both operands are false, (true AND true) will be true because both operands are true but the NOT preceding it will change it to false, and (true OR false) will be true because one operand is true. Then we have the statement true AND false AND true, which will be false because the AND needs all operands to be true to be true, but because the whole statement is preceded by a NOT, the final value is true.

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

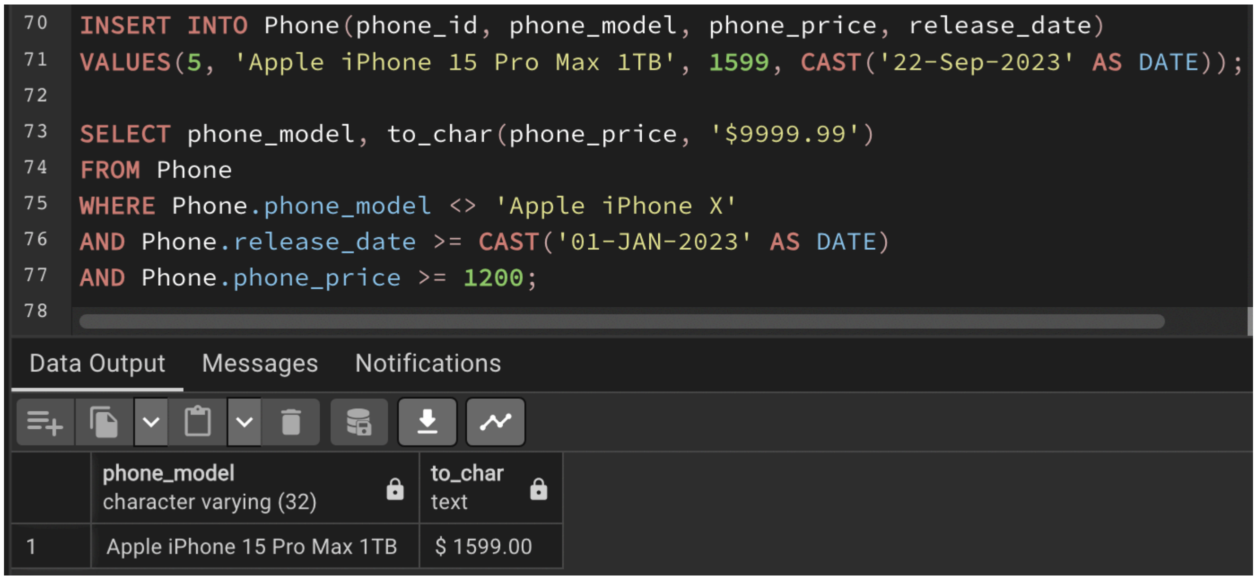
a. Any Phone matching the following condition is considered a high-end Phone for the store: Any Phone, except for the "Apple iPhone X" Phone, that is available on or after 05/01/2020, with a price of $900.00 or higher, is a high-end Phone. Write a query that shows the name and price of all high-end Phones.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



```sql
SELECT phone_model, to_char(phone_price, '$9999.99')
FROM Phone
WHERE Phone.phone_model <> 'Apple iPhone X'
AND Phone.release_date >= CAST('01-MAY-2020' AS DATE)
AND Phone.phone_price >= 900;
```

	phone_model character varying (32)	to_char text
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.



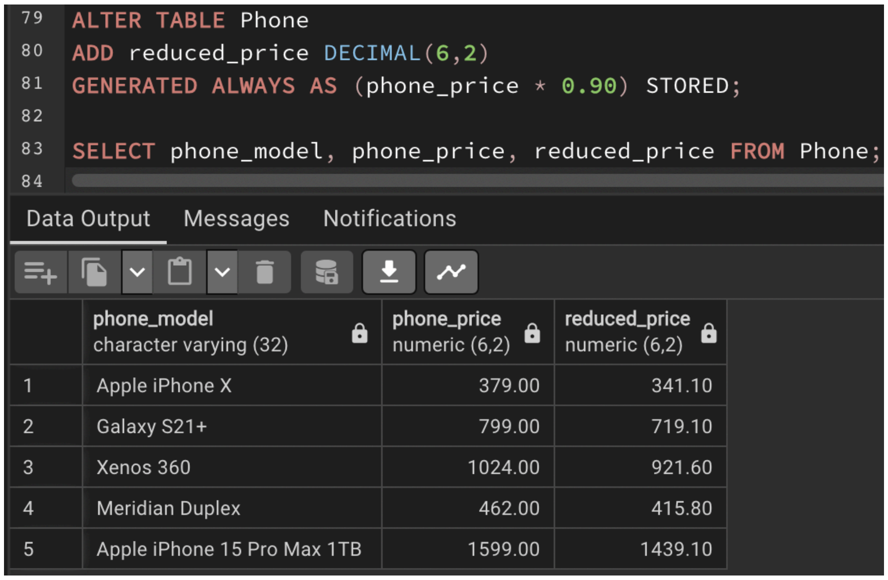
```
INSERT INTO Phone(phone_id, phone_model, phone_price, release_date)
VALUES(5, 'Apple iPhone 15 Pro Max 1TB', 1599, CAST('22-Sep-2023' AS DATE));

SELECT phone_model, to_char(phone_price, '$9999.99')
FROM Phone
WHERE Phone.phone_model <> 'Apple iPhone X'
AND Phone.release_date >= CAST('01-JAN-2023' AS DATE)
AND Phone.phone_price >= 1200;
```

Data Output Messages Notifications

	phone_model character varying (32)	to_char text
1	Apple iPhone 15 Pro Max 1TB	$ 1599.00

13. *Using Generated Columns* – Address the following.

a. 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.

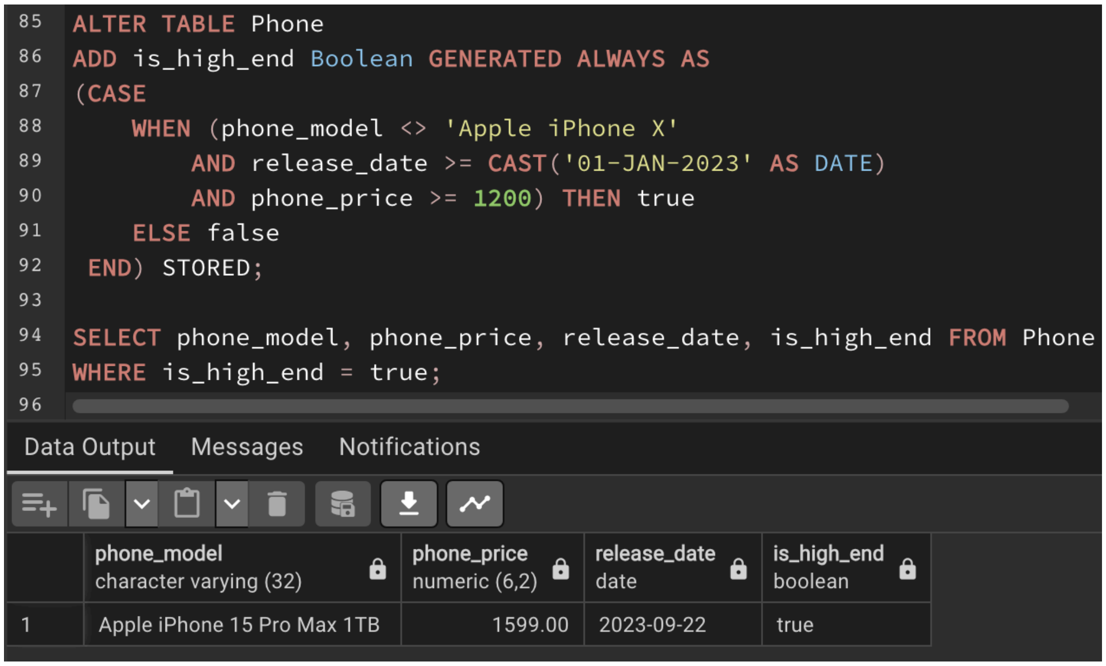


```
ALTER TABLE Phone
ADD reduced_price DECIMAL(6,2)
GENERATED ALWAYS AS (phone_price * 0.90) STORED;

SELECT phone_model, phone_price, reduced_price FROM Phone;
```

Data Output Messages Notifications

	phone_model character varying (32)	phone_price numeric (6,2)	reduced_price numeric (6,2)
1	Apple iPhone X	379.00	341.10
2	Galaxy S21+	799.00	719.10
3	Xenos 360	1024.00	921.60
4	Meridian Duplex	462.00	415.80
5	Apple iPhone 15 Pro Max 1TB	1599.00	1439.10

b. 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.



```sql
ALTER TABLE Phone
ADD is_high_end Boolean GENERATED ALWAYS AS
(CASE
    WHEN (phone_model <> 'Apple iPhone X'
        AND release_date >= CAST('01-JAN-2023' AS DATE)
        AND phone_price >= 1200) THEN true
    ELSE false
 END) STORED;

SELECT phone_model, phone_price, release_date, is_high_end FROM Phone
WHERE is_high_end = true;
```

	phone_model character varying (32)	phone_price numeric (6,2)	release_date date	is_high_end boolean
1	Apple iPhone 15 Pro Max 1TB	1599.00	2023-09-22	true