***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.

A screenshot of a computer

Description automatically generated

1. *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.

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

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

A screenshot of a computer code

Description automatically generated

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.

1. *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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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.

1. *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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. *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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. *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.

A screenshot of a computer

Description automatically generated

***Section Two – Expressing Data***

1. *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”).

A screenshot of a phone number

Description automatically generated

1. *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.

A screenshot of a computer

Description automatically generated

1. *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)

A screenshot of a computer

Description automatically generated

***Section Three – Advanced Data Expression***

1. *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.
   1. (true AND false) OR (false AND true)

False OR False

False

* 1. (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

* 1. 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

1. *Using Boolean Expressions in Queries* – Address the following scenarios.
   1. 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.

A screenshot of a computer

Description automatically generated

* 1. 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.

A screenshot of a computer

Description automatically generated

1. *Using Generated Columns –* Address the following.
   1. 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.

A screenshot of a phone number

Description automatically generated

* 1. 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.

A screenshot of a computer

Description automatically generated