|  |
| --- |
| **Overview of the Lab** |
| One purpose of this lab it to teach you how to meaningfully relate data and to answer questions using related data, using SQL. In the prior lab you learned the fundamentals creating and using tables to store data, which is a good introduction to SQL. To be effective, however, you need to know how to work with relationships that naturally occur in the data you work with.  We don’t need to look far to recognize examples of relationships. For just some examples, people have addresses, pets are owned by owners, products are sold by stores, and cases are heard in a court. Relationships are both plentiful and inevitable in virtually any database. In many ways, the richness and complexity of the relationships in a database determines its usefulness in answering important questions about the data.  It is likewise inevitable with virtually any database data items need be formatted into human readable form, or manipulated to derive different results. Directly extracting values exactly as they are stored in a database works for some but not all queries. For example, a customer table may store a first and a last name, such as “Smith” and “Bob”, but emails or letters to the customer would use the full name, “Bob Smith”. Another objective of this lab is for you to learn how to format and manipulate data using functions and expressions.  From a technical perspective, together, we will learn:   * how to enforce relationships between two tables using a FOREIGN KEY constraint. * how to add related data to related tables. * how to ask questions and answer them using SQL queries that relate data. * details about the significant components that determine how a value is displayed in your SQL client. * to understand and effectively use expressions to manipulate data values. * how to use formatting functions to format data values into to human readable or other formats. |

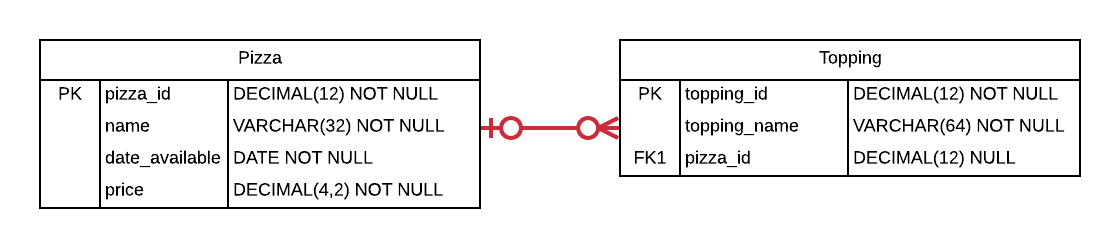
|  |
| --- |
| **Lab 2 Explanations Reminder** |
| As a reminder, it is important to read through the Lab 2 Explanation document to successfully complete this lab, available in the assignment inbox alongside this lab. The explanation document illustrates how to correctly execute each SQL construct step-by-step, and explains important theoretical and practical details. |

|  |
| --- |
| **Other Reminders** |
| * The examples in this lab will execute in modern versions of Oracle, Microsoft SQL Server, and PostgreSQL as is. * The screenshots in this lab display execution of SQL in the default SQL clients supported in the course – Oracle SQL Developer, SQL Server Management Studio, and pgAdmin – but your screenshots may vary somewhat as different version of these clients are released. * Don’t forget to commit your changes if you work on the lab in different sittings, using the “COMMIT” command, so that you do not lose your work. |

**Section One – Relating Data**

**Section Background**

To practice relating data, you will be working with the following simplified Pizza and Toppings schema.



In this schema, the Pizza table contains a primary key, the name of the pizza (for example, “Veggie” or “Meat Lovers”), the date when the pizza became available to order, and the price of the pizza. The Topping table contains a primary key, the name of the topping (such as “Sausage” or “Peppers”), and a foreign key that references the Pizza the topping is put on. The foreign key enforces the relationship between Pizza and Topping so that many toppings can be a part of a pizza. The foreign key is nullable since a particular topping may not have been assigned to a pizza (for example, perhaps a topping is available as an add-on but not part of the standard ingredients). There can also exist a pizza that has no toppings, namely, a plain pizza that only has tomato sauce and spice.

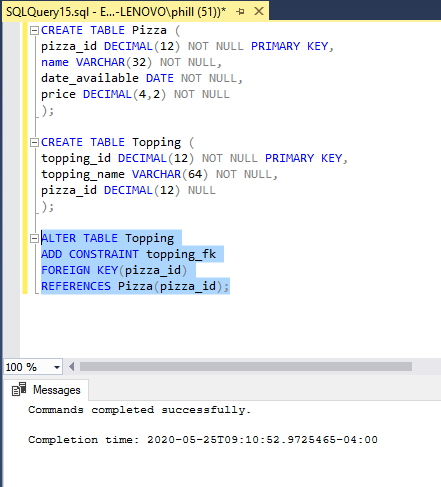
The schema is intentionally simplified when compared to what you might see in a real-world production schema. The schema only allows a particular topping to be a standard ingredient for one Pizza. The schema does not record a history of price changes as the price changes, nor does it support special pricing during special events. Many other attributes that would exist in a production database are not present. The current complexity is sufficient; additional complexity in the schema would not aid your learning at this point.

Do not worry if you don’t yet fully understand foreign keys and relationships. The Lab 2 explanations document gives you the information you need to complete the steps in this lab.

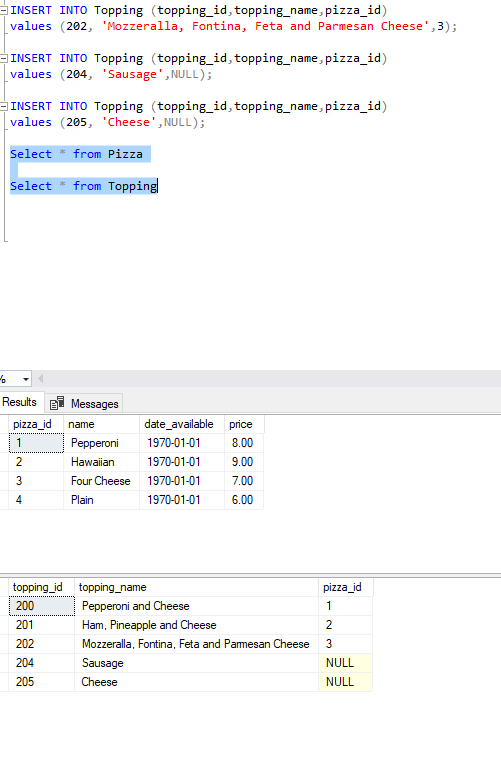
As a reminder, for each step that requires SQL, make sure to capture a screenshot of the command and the results of its execution.

**Section Steps**

1. Create the Pizza and Toppings tables, including all of their columns, datatypes, and constraints, including the foreign key constraint.



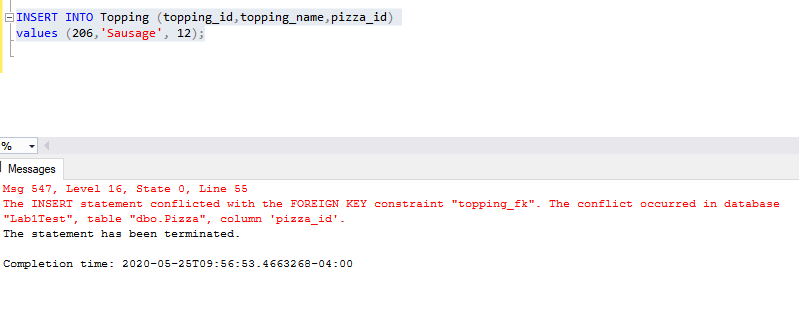
1. Insert at least four rows into the Pizza table. One of the pizzas should be named “Plain” because it will not have any toppings. Other than this, you select the ids, names, dates, and prices of your choosing (maybe you have some favorite pizzas?).   
     
   Insert toppings of your choosing into the topping table. One of the toppings should not be associated with any pizza, that is, the topping should be an “add-on” which is not included in any pizza’s standard toppings. Ensure that the “Plain” pizza has no toppings, and that all other pizzas have at least two toppings. Select all rows in both tables to view what you inserted.



1. As an exercise, attempt to insert a topping that references a pizza 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 Pizza reference is invalid.

The foreign key in the topping table is not present.

The foreign key used does not correspond to a primary key in the pizza table. The error indicates the conflict in the constraint.



1. Summarize:
   1. what a join is and how joins help answer questions using related data, and
   2. the similarities and differences between an inner join, a left join, a right join, and a full outer join.

A join is a way to retrieve values from a data table based on related tables.

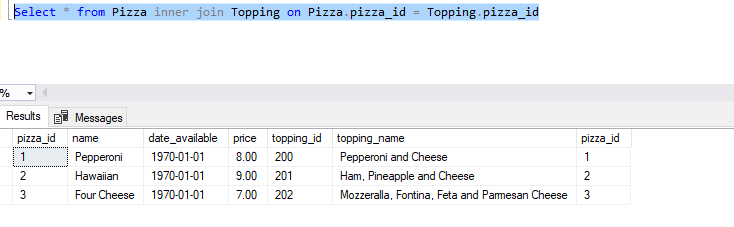
I view the differences between the joins by visualizing two intersecting circles.

Inner Outer (everything) Left

not this

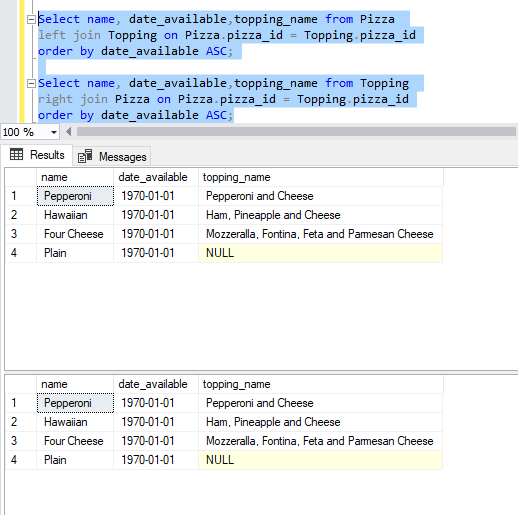
Right Join

1. With a single SQL query, fulfill the following request:  
     
   List the names of the pizzas that have toppings, and the names of all of the toppings that go with each pizza.

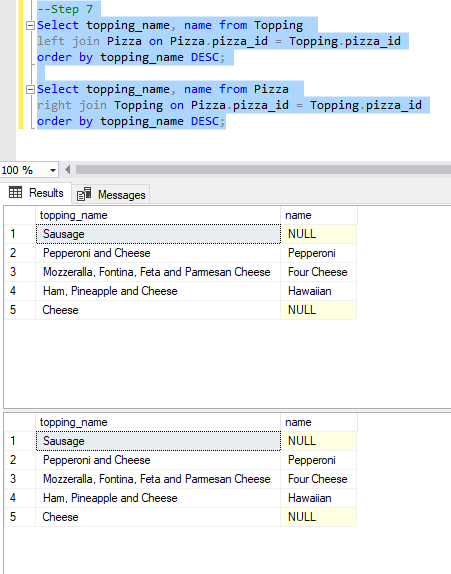
  
From a technical SQL perspective, explain why some rows in the Pizza table and some rows in the Toppings table were not listed.

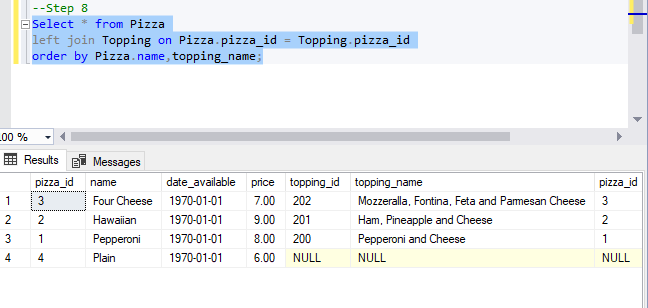
The intersection of the primary key from the Pizza table did not correspond to the Foreign Key in the Toppings table.

1. Fulfill the following request:  
     
   List the names and availability date of all pizzas whether or not they have toppings. For the pizzas that have toppings, list the names of the toppings that go with each of those pizzas. Order the list by the availability 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.



1. Fulfill the following request:  
     
   List the names of all toppings whether or not they go with a pizza, and the names of the pizzas the toppings go with. Order the list by topping name in reverse alphabetical order.  
     
   Just as with step #6, 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.



1. Fulfill the following request with a single SQL query:  
     
   List the names of all pizzas and all toppings, as well as which pizzas go with which toppings. Order the list alphabetically by pizza name then by topping name.
2. 

**Section Two – Expressing Data**

**Section Background**

While it is certainly useful to directly extract values as they are stored in a database, it is more useful in some contexts to manipulate these values to derive a different result. In this section we practice using value manipulation techniques to transform data values in useful ways. For example, what if we want to tell a customer exactly how much money they need to give for a purchase? We could extract a price and sales tax from the database, but it would be more useful to compute a price with tax as a single value by multiplying the two together and rounding appropriately, and formatting it as a currency, as illustrated in the figure below.

|  |  |
| --- | --- |
| *Less Useful to Customer* | |
| **price** | **tax\_percent** |
| 7.99 | 8.5 |

|  |
| --- |
| *More Useful to Customer* |
| **price\_with\_tax** |
| $8.67 |

We do not need to store the price with tax, because we can derive it when we need it.

As another example, what if we need to send an email communication to a customer by name? We could extract the prefix, first name, and last name of the customer, but it would be more useful to properly format the name by combining them in proper order, as illustrated below.

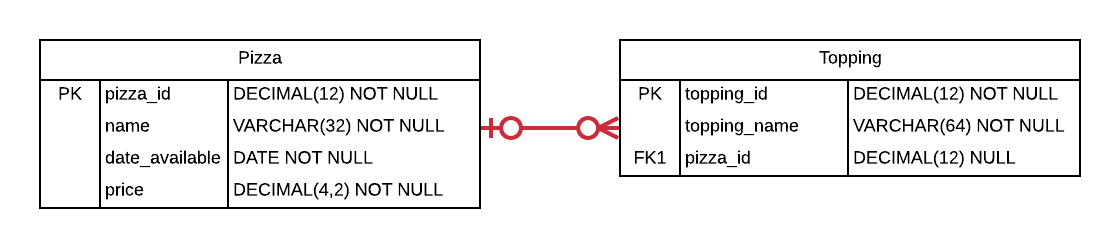
|  |  |  |
| --- | --- | --- |
| *Less Useful to Customer* | | |
| **prefix** | **first\_name** | **last\_name** |
| Mr. | Seth | Nemes |

|  |
| --- |
| *More Useful to Customer* |
| **name** |
| Mr. Seth Nemes |

Again, we do not need to store the formatted name, because we can derive it when we need it from its constituent parts. Manipulating raw data values stored in database tables can yield a variety of useful results we need without adding the burden of storing every such result.

In this section, you use expressions to manipulate and format data values. The first several steps in this section teach you several important concepts needed to correctly use expressions, including attributes of SQL clients, operator precedence, datatype precedence, and formatting functions. The later steps have you use this knowledge to manipulate and format data values.

You work with the same Pizza and Toppings schema from Section One. The schema is illustrated below again for your review.



**Section Steps**

1. Summarize in your own words why it’s not ideal to manually lookup or eyeball answers to questions about data in a database.

We need to create the logic to answer the questions about the data, not manually look up date ourselves. It should be consistent now and into the future.

1. Summarize in your own words four significant components that determine how a value is displayed in your SQL client.

Raw data type values in the DB.

Manipulations of the data type values.

Formatting – affects the appearance of the data

Client – Postgres, MS SQL Server, Toad all may display result differently.

1. Summarize in your own words:
   1. what expressions are.

Operands and operators that manipulate the data in tables.

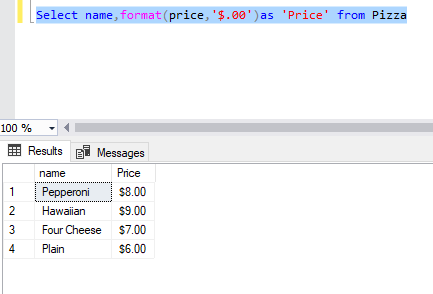
* 1. what operator precedence is and the role it plays in expressions.

Like mathematical precedence, these are DB rules that govern which operations occurs before others.

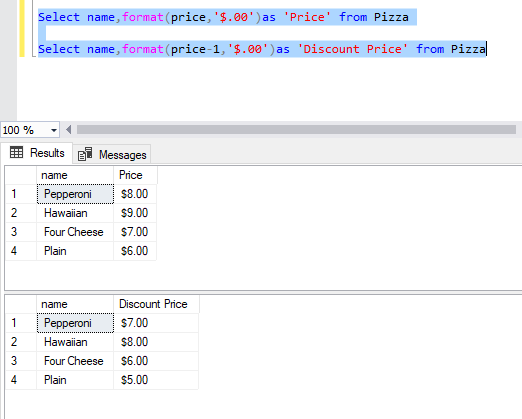
* 1. what datatype precedence is and the role it plays in expressions.

When mixing data types, the DBMS will usually choose the data type that has more ‘resolution’, more granularity – int is added to a floating point- the result will be a floating point.

1. Fulfill the following request with a single query:  
   Management of the pizza shop wants to review its pizza pricing. List the names and prices of all pizzas, making sure to format the price monetarily in U.S. dollars (for example, “$11.99”).



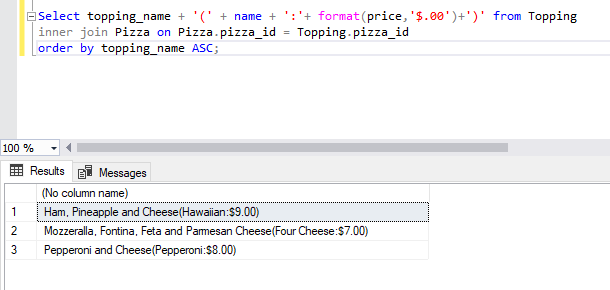
1. Fulfill the following request with a single query:  
     
   The pizza shop is running a special where every pizza is discounted by a dollar. List the names and discounted prices of all pizzas, making sure to format the price monetarily in U.S. dollars.



1. Fulfill the following request with a single query:

The pizza shop wants to mail out mailers that promotes all of the toppings it offers, tied into the pizzas it sells. The shop wants each line in the mailer formatted like “ToppingName (PizzaName: Price)”, and wants the lines ordered alphabetically by topping name. For example, if a “Meat Lover’s” pizza costs $10.00 and has two toppings – Sausage and Pepperoni – the results would have two lines for this pizza:  
  
Pepperoni (Meat Lover’s: $10.00)

Sausage (Meat Lover’s: $10.00)



# Evaluation

Your lab will be reviewed by your facilitator or instructor with the following criteria and grade breakdown.



Use the **Ask the Facilitators Discussion Forum** if you have any questions regarding how to approach this lab. Make sure to include your name in the filename and submit it in the *Assignments* section of the course.