Customers table:

*SELECT \* FROM customers;*

|  |  |  |
| --- | --- | --- |
| **first\_name**  **character varying (100)** | **id**  **integer** | **last\_name**  **character varying (255)** |
| Rolf | 1 | Smith |
| Jose | 2 | Salvatierra |
| Anne | 3 | Watson |
| Craig | 4 | Scott |
| Michael | 5 | Adam |

Items table:

*SELECT \* FROM items;*

|  |  |  |
| --- | --- | --- |
| **name**  **character varying (255)** | **id**  **integer** | **price**  **numeric (10,2)** |
| Pen | 1 | 5.00 |
| Fountain Pen | 2 | 11.30 |
| Laptop | 4 | 899.00 |
| Screen | 5 | 275.50 |
| Hard Drive | 6 | 89.99 |
| Ink | 3 | 5.00 |

Purchases’ table

*SELECT \* FROM purchases;*

|  |  |  |
| --- | --- | --- |
| **id**  **integer** | **item\_id**  **integer** | **customer\_id**  **integer** |
| 1 | 4 | 1 |
| 2 | 5 | 1 |
| 3 | 6 | 1 |
| 4 | 1 | 3 |
| 5 | 3 | 5 |
| 6 | 2 | 5 |
| 7 | 4 | 2 |
| 8 | 2 | 4 |
| 9 | 3 | 4 |
| 10 | 6 | 5 |

In this lesson we are looking at some SQL join examples. Notice that the data we got in our sample database is more complex that we saw in the lecture video. That is nice because it would let us perform JOINs with three tables.

The first thing we will do is build these queries step by step, so we are confident that we understand exactly all the different parts of the SQL query.

The first query that we got here is

*SELECT \* FROM purchases;*

|  |  |  |
| --- | --- | --- |
| **id**  **integer** | **item\_id**  **integer** | **customer\_id**  **integer** |
| 1 | 4 | 1 |
| 2 | 5 | 1 |
| 3 | 6 | 1 |
| 4 | 1 | 3 |
| 5 | 3 | 5 |
| 6 | 2 | 5 |
| 7 | 4 | 2 |
| 8 | 2 | 4 |
| 9 | 3 | 4 |
| 10 | 6 | 5 |

The purchases table gives us purchase ID, item ID and customer ID. The purchases’ table lets us get the item that was bought and the customer who bought the item.

*SELECT \* FROM items;*

|  |  |  |
| --- | --- | --- |
| **name**  **character varying (255)** | **id**  **integer** | **price**  **numeric (10,2)** |
| Pen | 1 | 5.00 |
| Fountain Pen | 2 | 11.30 |
| Laptop | 4 | 899.00 |
| Screen | 5 | 275.50 |
| Hard Drive | 6 | 89.99 |
| Ink | 3 | 5.00 |

The items’ table has the items’ names, the items’ ids and the items’ prices.

*SELECT \* FROM customers;*

|  |  |  |
| --- | --- | --- |
| **first\_name**  **character varying (100)** | **id**  **integer** | **last\_name**  **character varying (255)** |
| Rolf | 1 | Smith |
| Jose | 2 | Salvatierra |
| Anne | 3 | Watson |
| Craig | 4 | Scott |
| Michael | 5 | Adam |

The customers’ table has the first name, the last name and as well as the customers’ id.

The IDs in all these tables are used to link all customers with their items and their purchases in the purchases’ table. This means if we want to get those items that have been bought then we could do a JOIN from items to purchases and find out which purchases are associated with which items.

What type of JOIN we should use for this case? Should it be OUTER JOIN (LEFT JOINT), or should it be a RIGHT JOIN, or should it be a FULL JOIN, or should it be an INNER JOIN?

The answer is INNER JOIN.

That is because we want to get the items that has purchases associated with them, and we do not want to get the items that has no purchases.

*SELECT \* FROM items INNER JOIN purchases*

*ON customer.id = purchases.id;*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **name**  **character varying (255)** | **id**  **integer** | **price**  **numeric (10,2)** | **id**  **integer** | **item\_id**  **integer** | **customer\_id**  **integer** |
| Screen | 5 | 275.50 | 2 | 5 | 1 |
| Hard Drive | 6 | 89.99 | 3 | 6 | 1 |
| Ink | 3 | 5.00 | 5 | 3 | 5 |
| Fountain Pen | 2 | 11.30 | 6 | 2 | 5 |
| Fountain Pen | 2 | 11.30 | 8 | 2 | 4 |
| Ink | 3 | 5.00 | 9 | 3 | 4 |
| Hard Drive | 6 | 89.99 | 10 | 6 | 5 |

In our resultant table we get items’ details linked up with our purchases’ details.

If instead of an INNER JOIN we do a LEFT JOIN, what do you think is going to happen here?

The answer is we are going to get items that has no purchases with also the items that has purchases in our resultant table.

*SELECT \* FROM items LEFT JOIN purchases*

*ON items.id = purchases.item\_id;*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **name**  **character varying (255)** | **id**  **integer** | **price**  **numeric (10,2)** | **id**  **integer** | **item\_id**  **integer** | **customer\_id**  **integer** |
| Screen | 5 | 275.50 | 2 | 5 | 1 |
| Hard Drive | 6 | 89.99 | 3 | 6 | 1 |
| Ink | 3 | 5.00 | 5 | 3 | 5 |
| Fountain Pen | 2 | 11.30 | 6 | 2 | 5 |
| Fountain Pen | 2 | 11.30 | 8 | 2 | 4 |
| Ink | 3 | 5.00 | 9 | 3 | 4 |
| Hard Drive | 6 | 89.99 | 10 | 6 | 5 |
| Pen | 1 | 5.00 |  |  |  |

So, here we did a LEFT JOIN and get all the items those are associated with a purchase and the item that has no purchase with purchase’s field blank for that item.

We can get the same outcome by using a RIGHT JOIN, but in the other way around, by getting all purchases those are associated with their items.

*SELECT \* FROM purchases*

*RIGHT JOIN items ON items.id = purchases.item\_id;*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id  integer | item\_id  integer | customer\_id  character varying (255) | name  character varying (255) | id  integer | price  numeric |
| 2 | 5 | 1 | Screen | 5 | 275.50 |
| 3 | 6 | 1 | Hard Drive | 6 | 89.99 |
| 5 | 3 | 5 | Ink | 3 | 5.00 |
| 6 | 2 | 5 | Fountain Pen | 2 | 11.30 |
| 8 | 2 | 4 | Fountain Pen | 2 | 11.30 |
| 9 | 3 | 4 | Ink | 3 | 5.00 |
| 10 | 6 | 5 | Hard Drive | 6 | 89.99 |
|  |  |  | Pen | 1 | 5.00 |

The performance of this RIGHT JOIN is identical to the LEFT JOIN, but it is just showing the purchases before items and other than that everything is pretty much the same and we can do it the either way.

Now, lets find out which customers did purchases.

*SELECT \* FROM customers INNER JOIN purchases*

*ON customers.id = purchases.customer\_id;*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) | id  integer | item\_id  integer | customer\_id  integer |
| Rolf | 1 | Smith | 2 | 5 | 1 |
| Rolf | 1 | Smith | 3 | 6 | 1 |
| Michael | 5 | Adam | 5 | 3 | 5 |
| Michael | 5 | Adam | 6 | 2 | 5 |
| Craig | 4 | Scott | 8 | 2 | 4 |
| Craig | 4 | Scott | 9 | 3 | 4 |
| Michael | 5 | Adam | 10 | 6 | 5 |

We can also try a LEFT JOIN that will give us the customers’ data and then give us the purchase data with it, including the customers that never made any purchase, leaving their purchases’ fields blank.

*SELECT \* FROM customers LEFT JOIN purchases*

*ON customers.id = purchases.customer\_id;*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) | id  integer | item\_id  integer | customer\_id  integer |
| Rolf | 1 | Smith | 2 | 5 | 1 |
| Rolf | 1 | Smith | 3 | 6 | 1 |
| Michael | 5 | Adam | 5 | 3 | 5 |
| Michael | 5 | Adam | 6 | 2 | 5 |
| Craig | 4 | Scott | 8 | 2 | 4 |
| Craig | 4 | Scott | 9 | 3 | 4 |
| Michael | 5 | Adam | 10 | 6 | 5 |
| Jose | 2 | Salvatierra |  |  |  |
| Anne | 3 | Watson |  |  |  |

Let’s now see how we can only get the customers’ data that have made purchases, but we only want the customers data that made purchases and not the purchase data.

*SELECT customers.first\_name, customers.last\_name FROM customers*

*INNER JOIN purchases*

*ON customers.id = purchases.customers\_id;*

|  |  |
| --- | --- |
| first\_name  character varying (100) | last\_name  character varying (255) |
| Rolf | Smith |
| Rolf | Smith |
| Michael | Adam |
| Michael | Adam |
| Craig | Scott |
| Craig | Scott |
| Michael | Adam |

Now we get the names of customers for each of the purchases they made, but not the purchases’ data. As some customers made multiple purchases, that is why their names occurred multiple times.

Now we will see how we can perform JOINs using three tables simultaneously.

*SELECT \* FROM items*

*INNER JOIN purchases ON items.id = purchases.item\_id;*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **name**  **character varying (255)** | **id**  **integer** | **price**  **numeric** | **id**  **integer** | **item\_id**  **integer** | **customer\_id**  **integer** |
| Screen | 5 | 275.50 | 2 | 5 | 1 |
| Hard Drive | 6 | 89.99 | 3 | 6 | 1 |
| Ink | 3 | 5.00 | 5 | 3 | 5 |
| Fountain Pen | 2 | 11.30 | 6 | 2 | 5 |
| Fountain Pen | 2 | 11.30 | 8 | 2 | 4 |
| Ink | 3 | 5.00 | 9 | 3 | 4 |
| Hard Drive | 6 | 89.99 | 10 | 6 | 5 |

The resultant table is displaying the items and the purchases those are associated with these items. The pen item has got disappeared from our table as pen was never purchased by any customer. Some of the items are being displayed multiple times as they were bought multiple times by customers.

Each of these items has a customer id at the last column showing each of the customer that bought these items. Now, we can join this resultant table with customers’ table. Do remember that we are joining the resultant table with the customers’ table. We are not joining customers’ table with purchases’ table; we are not joining customers’ table with items’ table. We are performing the joining of items’ table with purchases’ table first and then we are joining customers’ table with it. That is why in the join below, we already have access to *purchases.customer\_id.* Because that has already been joined before.

*SELECT \* FROM items*

*INNER JOIN purchases ON items.id = purchases.item\_id*

*INNER JOIN customers ON purchases.customer\_id = customers.id;*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **name**  **character varying (255)** | **id**  **integer** | **price**  **numeric (10,2)** | **id**  **integer** | **item\_id**  **integer** | **customer\_id**  **integer** | **first\_name**  **character varying (100)** | **id**  **integer** | **last\_name**  **character varying (255)** |
| Screen | 5 | 275.50 | 2 | 5 | 1 | Rolf | 1 | Smith |
| Hard Drive | 6 | 89.99 | 3 | 6 | 1 | Rolf | 1 | Smith |
| Ink | 3 | 5.00 | 5 | 3 | 5 | Michael | 5 | Adam |
| Fountain Pen | 2 | 11.30 | 6 | 2 | 5 | Michael | 5 | Adam |
| Fountain Pen | 2 | 11.30 | 8 | 2 | 4 | Craig | 4 | Scott |
| Ink | 3 | 5.00 | 9 | 3 | 4 | Craig | 4 | Scott |
| Hard Drive | 6 | 89.99 | 10 | 6 | 5 | Michael | 5 | Adam |

In the resultant table we get the same data as before up to customers’ id and then we get our customers’ data, starting with customers’ first names joined into it.

If we want, we can get specific columns at this relation instead of getting all the columns and jumbling up our resultant table.

For example, we only want to see the customers that made purchases, with what items they bought and what were the prices of those items. In that case we will write our query as below.

*SELECT customers.first\_name, customers.last\_name, items.name, items.price FROM items*

*INNER JOIN purchases ON items.id = purchases.item\_id*

*INNER JOIN customers ON purchases.customer\_id = customers.id;*

|  |  |  |  |
| --- | --- | --- | --- |
| **first\_name**  **character varying (100)** | **last\_name**  **character varying (255)** | **name**  **character varying (255)** | **price**  **numeric (10,2)** |
| Rolf | Smith | Screen | 275.50 |
| Rolf | Smith | Hard Drive | 89.99 |
| Michael | Adam | Ink | 5.00 |
| Michael | Adam | Fountain Pen | 11.30 |
| Craig | Scott | Fountain Pen | 11.30 |
| Craig | Scott | Ink | 5.00 |
| Michael | Adam | Hard Drive | 89.99 |