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 GROUP BY.

GROUP BY is going to be fantastic, because it is going to let us remove duplicates when we perform JOINs, it is going to let us make sure that we are grouping our data and only getting the data related to the customers that we are interested in.

Here how it is going to work, we are going to do a JOIN and the we are going to get a bunch of duplicate data and then we are going to GROUP. Then the duplicate is going to get removed but naturally we are going to lose some granularity.

Let’s look this on an example,

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 |

We can see here our list of customers, and now we will do a LEFT JOIN on purchases from this table. That is going to give us purchases entries for all the customers. Customers who have not make any purchases their entries are going stay empty.

SELECT \* FROM customers LEFT JOIN purchases ON customers.id = purchases.customers\_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 |  |  |  |

As we can see some customers made multiple purchases, which is why we get multiple entries for them, thus their names appearing multiple times. Let’s say we are not interested on the purchase data for each of the purchases each customer made. We just want to know how many purchases each individual customer made.

That is why, first we will just get the purchase ids for each purchases the customers made.

SELECT customers.first\_name, customers.last\_name, purchases.id FROM customers LEFT JOIN purchases ON customers.id = purchases.customers\_id;

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

Here we got all the purchases with their ids made by each customer. Now we can add a function called COUNT() around the purchase.id which will count each of the purchases made by each customers. But we need to add the GROUP BY to our query to eliminate the duplicate names.

SELECT customers.first\_name, customers.last\_name, COUNT(purchases.id) FROM customers LEFT JOIN purchases ON customers.id = purchases.customers\_id GROUP BY customers.id;

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | last\_name  character varying (255) | count  bigint |
| Craig | Scott | 2 |
| Michael | Adam | 3 |
| Rolf | Smith | 2 |
| Jose | Salvatierra | 0 |
| Anne | Watson | 0 |

Now, you can see that we get the names of each customer exactly 1 time, and the COUNT column at the end shows total how many purchases each of those customers made.

This adds a lot more value to what we can do with SQL, because now we can get sanitized and transformed information from SQL, its no longer just data. Its now turning into information.

If we want to do a normal GROUP BY. Then we can do that with only our customers to see the list of our customers, not linking it with any purchases or orders.

SELECT \* FROM customers GROUP BY customers.id;

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

Now the function we want to us is called SUM. It is to figure out how much money each of our customer has spent.

SELECT \* FROM items;

|  |  |  |
| --- | --- | --- |
| name  character varying (100) | id  integer | price  numeric (10,2) |
| Fountain Pen | 2 | 11.30 |
| Screen | 5 | 275.50 |
| Hard Drive | 6 | 89.99 |
| Pen | 1 | 5.00 |
| Ink | 3 | 5.00 |

SELECT \* FROM items

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| name  character varying (100) | 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 |

SELECT \* FROM items

INNER JOIN purchases ON items.id = purchases.items\_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 |

Now we need to make sure that there is only one customer per row and we need to add the GROUP BY to the customer id.

SELECT \* FROM items

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

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

GROUP BY customers.id;

ERROR

This query gives us ERROR because when we are using GROUP BY in a query we cannot SELECT \*, because we have grouped the rows and that is why for the names of items on the rows of item column we have multiple items purchased by one customer and this is the case for some customers and that is why when we are using GROUP BY the each name of customers will be displayed one time, so when they if they have bought multiple items we will not be able to display the second or third item we cannot use the customer’s name on the customers’ column more than one time.

That is why we can put the ITEMS column in the COUNT function and that will show how many items each of the customers bought.

SELECT customers.first\_name, customers.last\_name, COUNT(items.name) FROM items

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

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

GROUP BY customers.id;

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | last\_name  character varying (255) | count  bigint |
| Michael | Adam | 3 |
| Rolf | Smith | 2 |
| Craig | Scott | 2 |

We can also use the SUM function on the price of each items and that adds the prices of each items together.

SELECT customers.first\_name, customers.last\_name, SUM(items.price) FROM items

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

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

GROUP BY customers.id;

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | last\_name  character varying (255) | count  bigint |
| Michael | Adam | 106.29 |
| Rolf | Smith | 365.49 |
| Craig | Scott | 16.30 |

This gives us the total amount each customer spent.

We can also calculate the total amount spent by all customers.

SELECT \* FROM purchases;

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

SELECT \* FROM purchases INNER JOIN items on purchases.item\_id = items.id;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id  integer | item\_id  integer | customer\_id  integer | name  character varying (255) | id  integer | price  numeric (10,2) |
| 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 |

SELECT SUM(items.price) FROM purchases INNER JOIN items on purchases.item\_id = items.id;

|  |
| --- |
| sum  numeric |
| 488.08 |