In this lesson we are looking at ORDER BY which is another SQL clause. In order to look at some use for this clause, we are going to back and find out how much each customer has spent in our business.

SELECT \* FROM items

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

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| name  character varying (255) | id  integer | price  numeric | 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 |

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

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

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

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | last\_name  character varying (255) | price  numeric |
| Rolf | Smith | 275.50 |
| Rolf | Smith | 89.99 |
| Michael | Adam | 5.00 |
| Craig | Scott | 11.30 |
| Craig | Scott | 5.00 |
| Michael | Adam | 89.99 |

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

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

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

GROUP BY customers.id;

ERROR

We get error because we cannot use GROUP BY command as long as we have the *items.price* column as there are specific customers who made multiple purchases and that is why we will use it to show us the total amount of purchase each customer made by using the SUM function.

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

INNER JOIN purchases ON purchases.item\_id = 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) | sum  numeric |
| Michael | Adam | 106.29 |
| Rolf | Smith | 365.49 |
| Craig | Scott | 16.30 |

And now we can order this SUM column using the ORDER BY command, from displaying the customer who spent least at the top to the customer who spent the most at the bottom.

SELECT customers.first\_name, customers.last\_name, SUM(items.price) AS “total\_spent” FROM items

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

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

GROUP BY customers.id

ORDER BY total\_spent;

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

If we wanted to ORDER BY the other way putting the customer who spent the highest amount at the top to the customer to spent the lowest amount at the bottom, then we have to include the DESC command which stands for descending, thus changing the order of the customers.

SELECT customers.first\_name, customers.last\_name, SUM(items.price) AS “total\_spent” FROM items

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

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

GROUP BY customers.id

ORDER BY total\_spent DESC;

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