It should open the Order table, then go though all the rows in the table, then aggregate the number of rows in total to give a value.

```
1  Use salesordersexample2;
2  EXPLAIN EXTENDED
3  SELECT COUNT(*) FROM Orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Orders	index	NULL	CustomerID	5	NULL	944	100.00	Using index

Here there was a simple query that was run on the table Orders. Its type was “Index” and extra was “using index” which means that here, the index is a covering index and can be used to satisfy all the data required from the table. Thus only the index tree is scanned. (this is faster than “all” because the size of index is smaller than table data).

Here the name of index or key is CustomerID and is of 5 characters/digits in length and would need possibly to look at 944 rows.

This is similar to what I had thought before, but there is one difference that stands out. Here the database didn’t look though the entire data in the table. Instead it used the primary key as an index can used that to find the number of rows.

This strategy was chosen as CustomerID is the primary key for that table and can uniquely identify the rows. Thus by using it as an index, the DBMS doesn’t need to look though all the data in the rest of the columns and thus is far faster and efficient.

(Ref: https://dev.mysql.com/doc/refman/8.0/en/explain-output.html#explain-join-types)