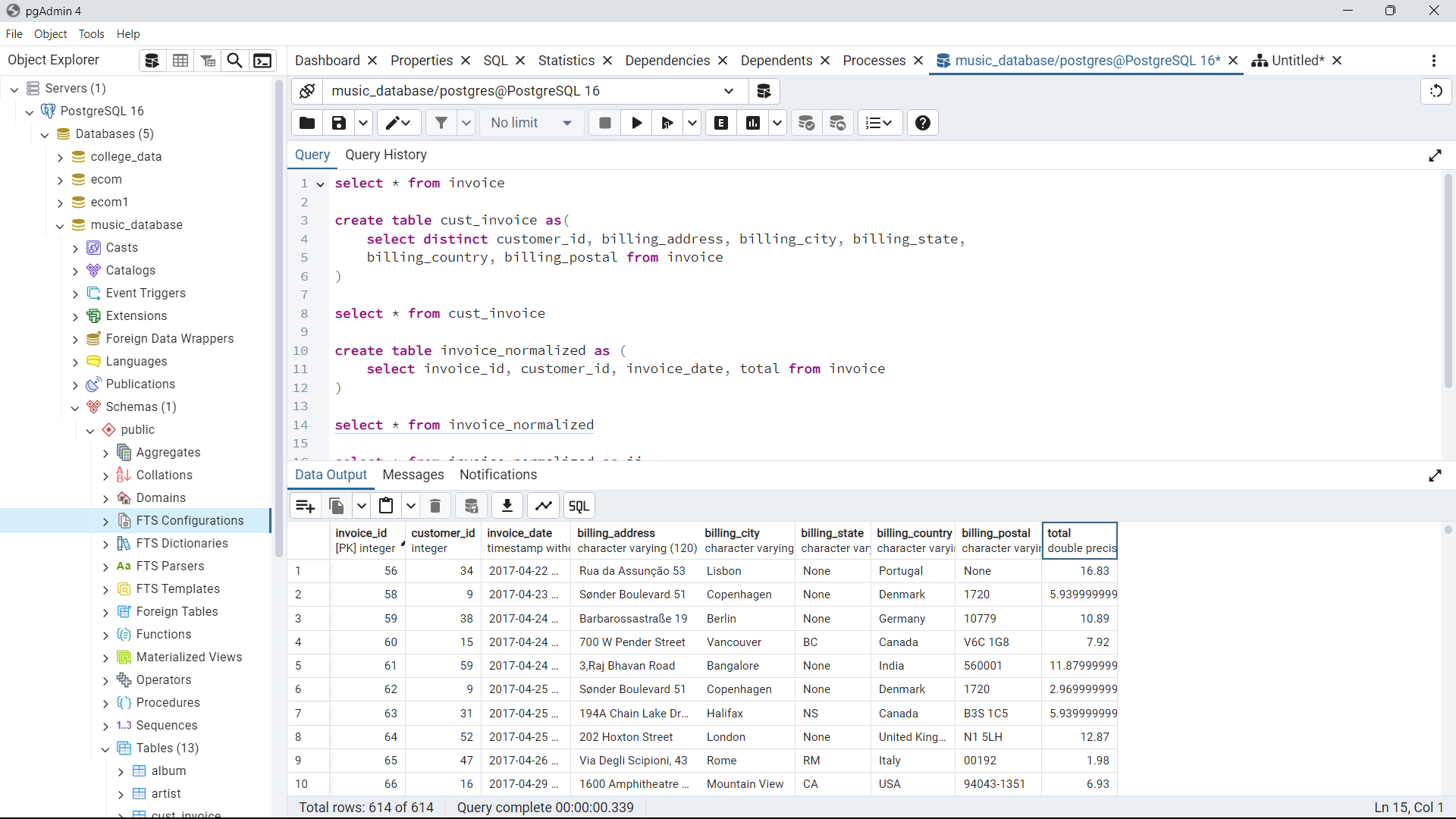
**Normalization of Invoice**

Introduction to Normalization

Normalization is a process in database design that organizes tables to minimize redundancy and dependency. It involves dividing large tables into smaller ones and defining relationships between them. The process ensures data integrity and makes it easier to manage and query data. There are several normal forms, including 1NF, 2NF, and 3NF, each addressing specific issues in database structure.

Original Data Structure

The original `Invoice` table contains information about invoice\_id, customer\_id, invoice\_date, billing\_address, postal and total in a single table. This table is not normalized, leading to potential data anomalies.



1NF (First Normal Form):

- Given table already in First Normal Form.

- All columns contain atomic values (no multiple values per column).

2NF (Second Normal Form):

2NF requires that the table is in 1NF and all non-key attributes are fully functional dependent on the primary key. We removed partial dependencies by creating a separate `cust\_invoice` table.

select \* from invoice

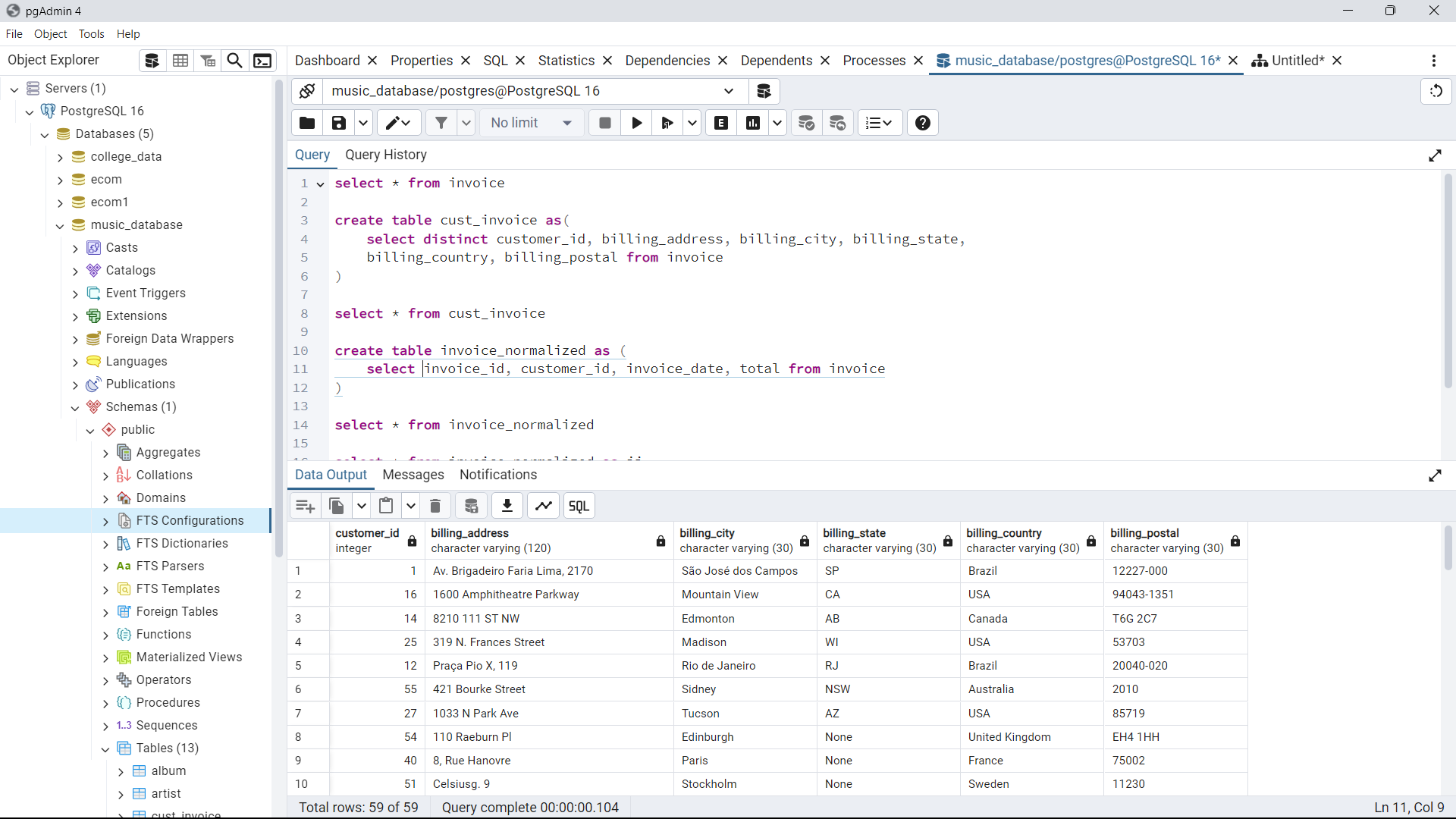
create table cust\_invoice as(

select distinct customer\_id, billing\_address, billing\_city, billing\_state,

billing\_country, billing\_postal from invoice

)

select \* from cust\_invoice



3NF (Third Normal Form):

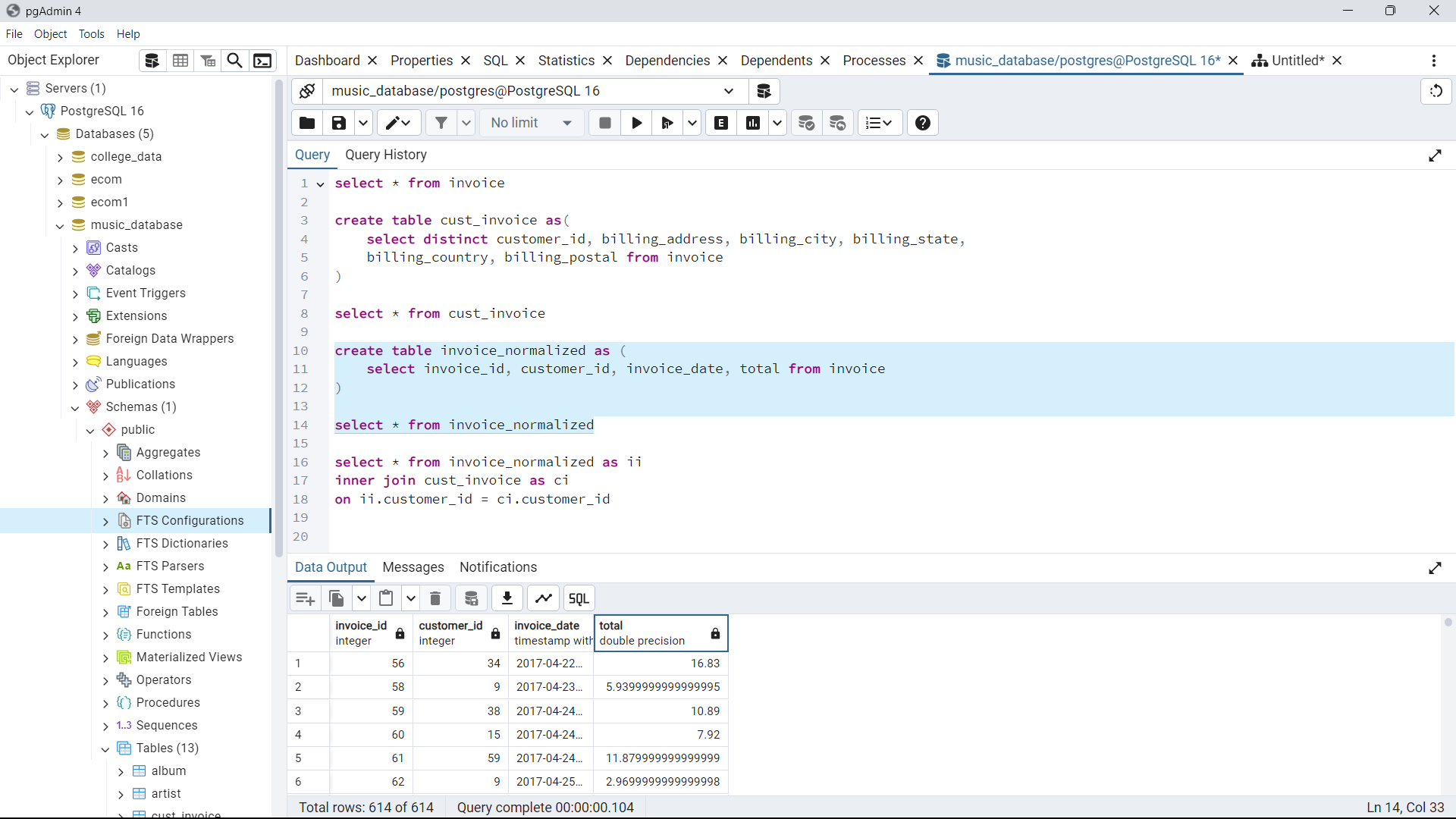
3NF requires that the table is in 2NF and all attributes are dependent only on the primary key. We removed transitive dependencies by creating a separate ` invoice\_normalized` table.

create table invoice\_normalized as (

select invoice\_id, customer\_id, invoice\_date, total from invoice

)

select \* from invoice\_normalized

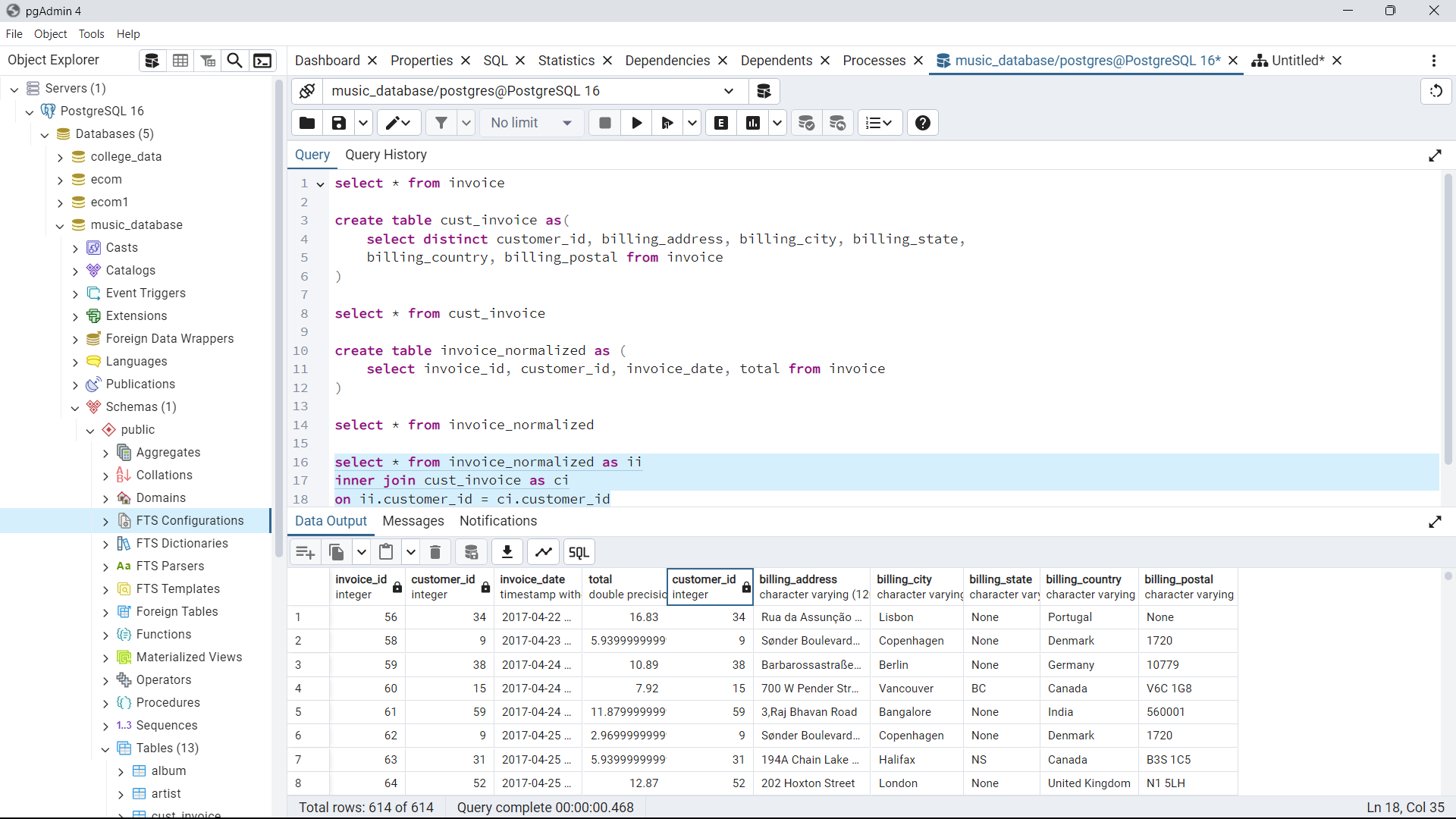


Join the tables using Inner Join:

select \* from invoice\_normalized as ii

inner join cust\_invoice as ci

on ii.customer\_id = ci.customer\_id



ERD for Database:

