**Rational Database Design Report**

1. **Business Background**

**Superstore Background:**

* **Superstore operates as an e-commerce platform, offering a broad spectrum of product categories throughout the United States.**
* **With the steady growth in their business volume, their current data storage approach is proving insufficient.**

**Superstore Demands:**

* **As data volumes surge, they require a more efficient storage and retrieval system.**
* **They also need user-friendly mechanisms for daily operations and in-depth analysis.**

**Project Objectives:**

* **Construct a relational database for Superstore that:**
* **Adheres to 3NF standards**
* **Preserves the original data**
* **Streamlines the data structure**
* **Facilitates flexible querying**

**Original Data Structure:**

* **Format: Stored in an Excel spreadsheet.**
* **Organization: All data is amalgamated within a single sheet.**
* **Row Significance: Each row denotes a distinct item within a particular order.**
* **Data Arrangement: Columns are diversified to describe attributes like product details, order date, and customer information.**

1. **Creating ER Model**

**Entities:**

* **Within the original data, there are 5 distinct information clusters.**
* **In the relational database, these clusters can be represented by 5 entities:**
* **Product Entity: Product details**
* **Order Entity: Order date**
* **Customer Entity: Order placer**
* **Location Entity: Shipping address**
* **Order\_item Entity: Order amount**

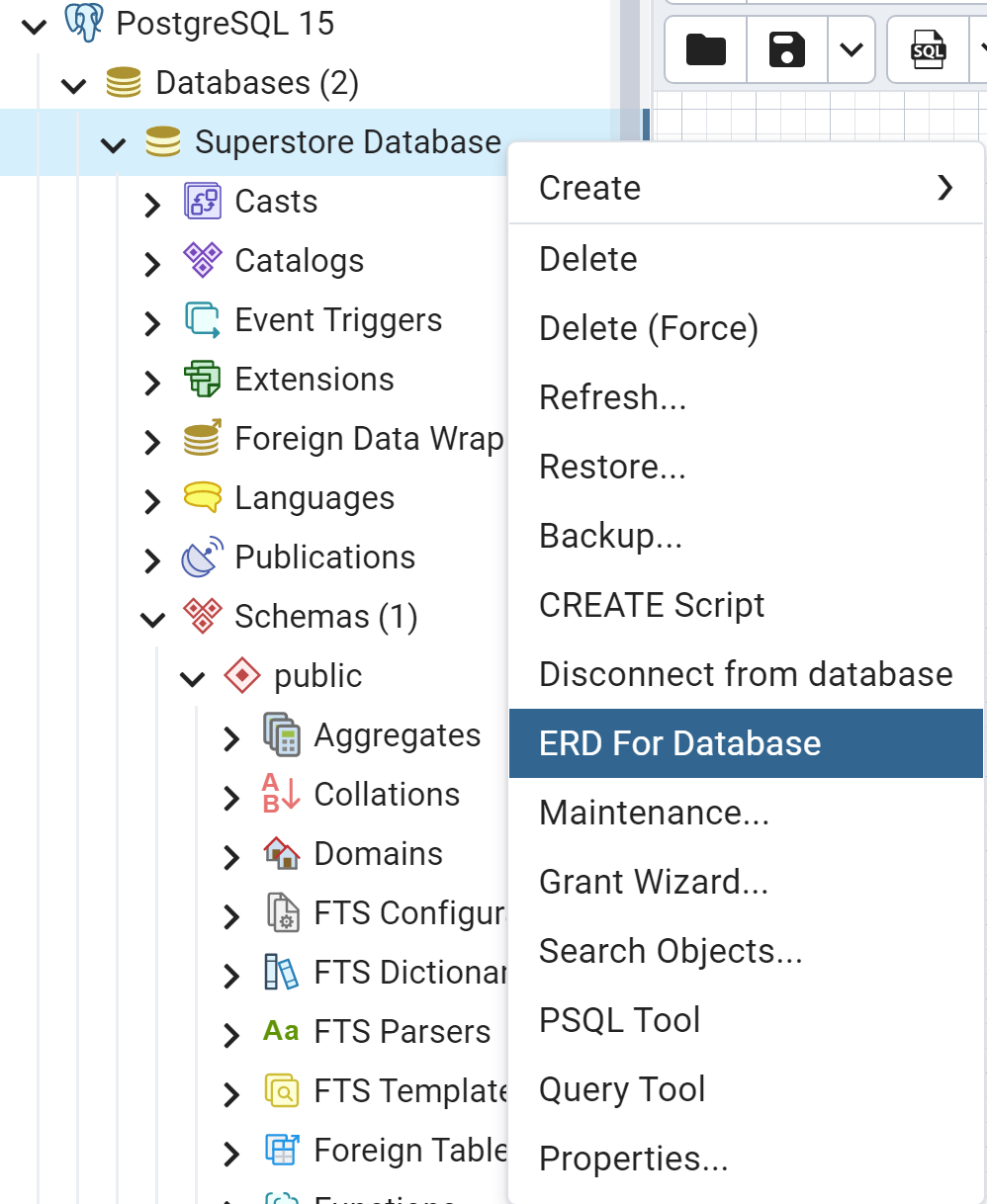
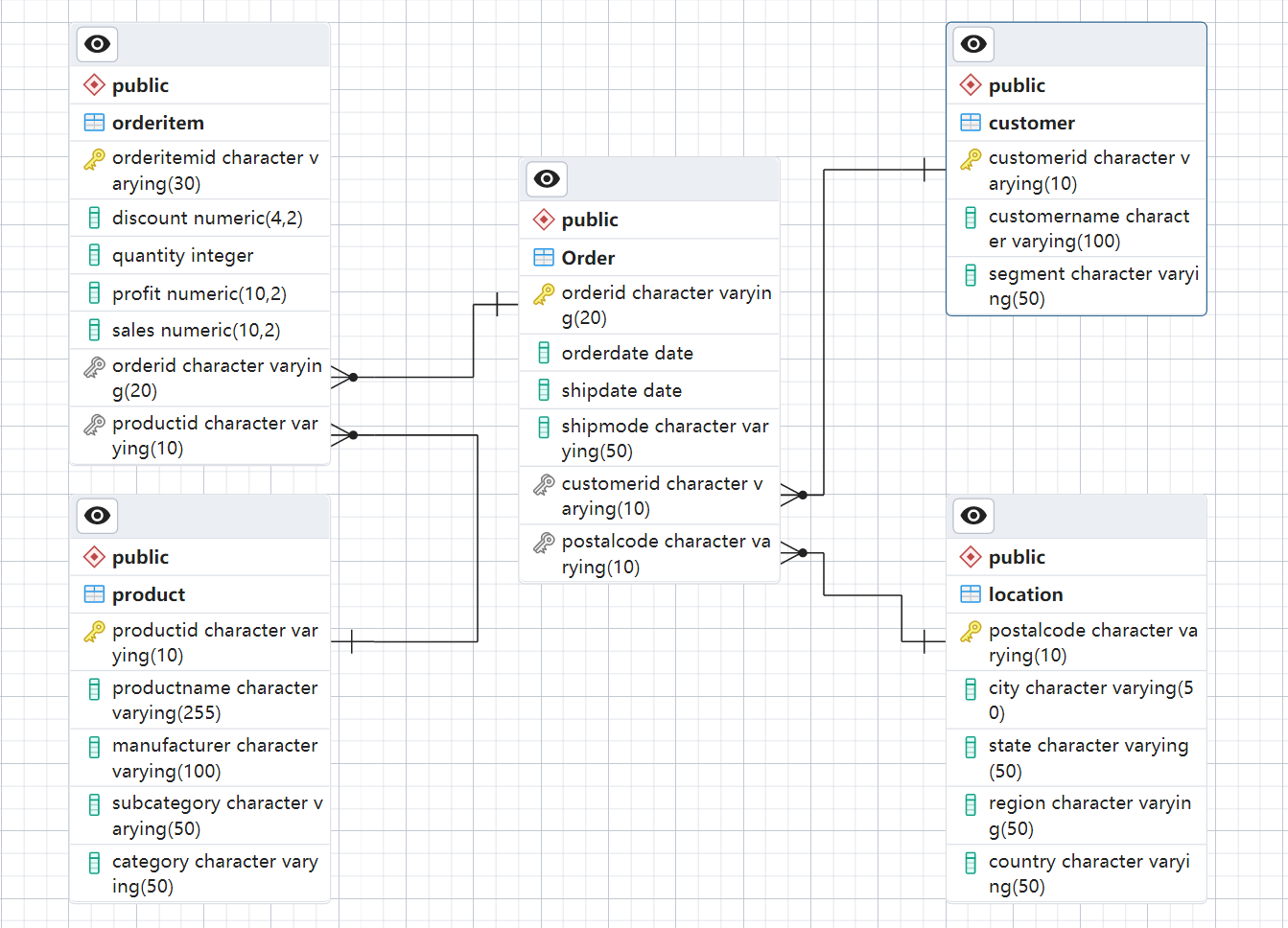
**Assumption and Keys:**

* **Product to Order\_item Relationship:**
* **A Product may belong to one or more Orders\_items; Each Order\_item must have one and only one Product.**
* **Product.ProductID (PK) --> Order\_item.ProductID (FK)**
* **Order to Order\_item Relationship:**
* **An Order may have one or more Orders\_items; Each Order\_item must belong to one and only one Order.**
* **Order.OrderID (PK) --> Order\_item.OrderID (FK)**
* **Customer to Order Relationship:**
* **A Customer may have one or more Orders; Each Order must belong to one and only one Customer.**
* **Customer.CustomerID (PK) --> Order.CustomerID (FK)**
* **Location to Order Relationship:**
* **A PostalCode may belong to one or more Orders; Each Order must have one and only one PostalCode.**
* **Location.PostalCode (PK) --> Order.PostalCode (FK)**

1. **Creating ERD**

**The ERD is constructed using PostgreSQL.**

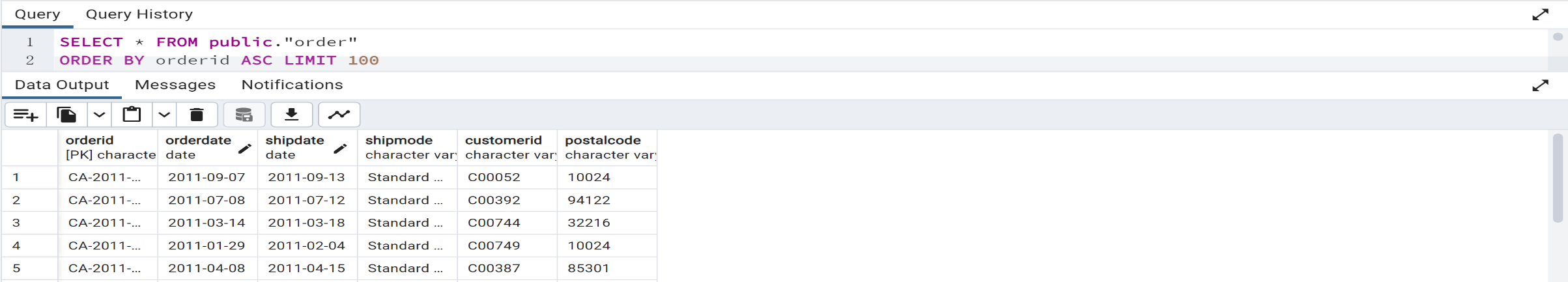
* **The 5 tables are interlinked using primary and foreign key relations.**



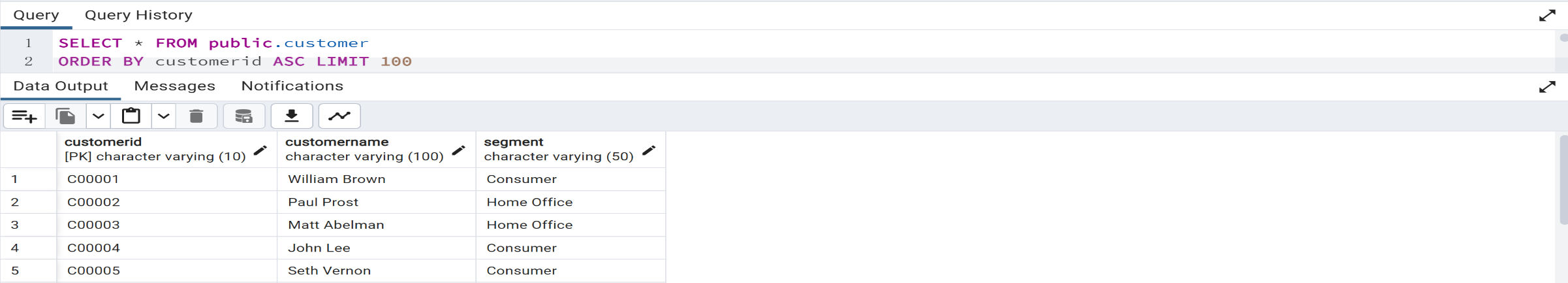
1. **Creating Relational DB**

**Five tables have been established using PostgreSQL.**

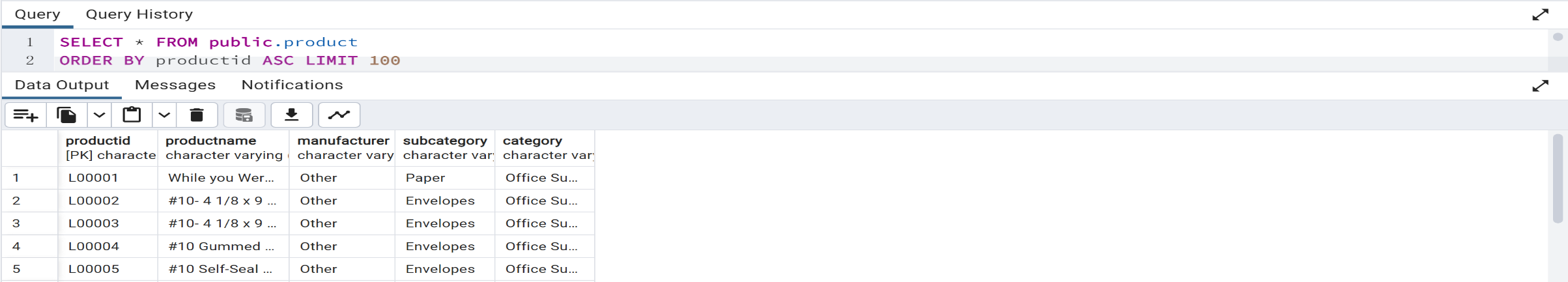
* **Order Table:**

****

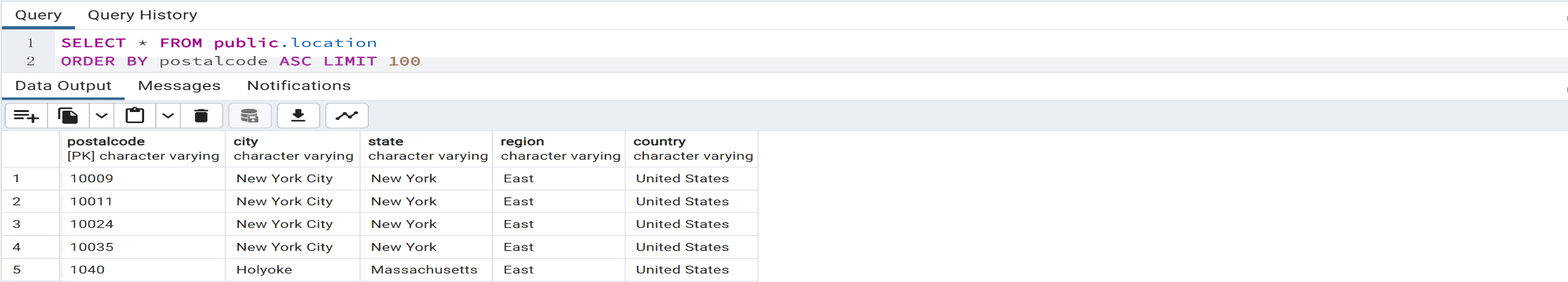
* **Customer Table:**

****

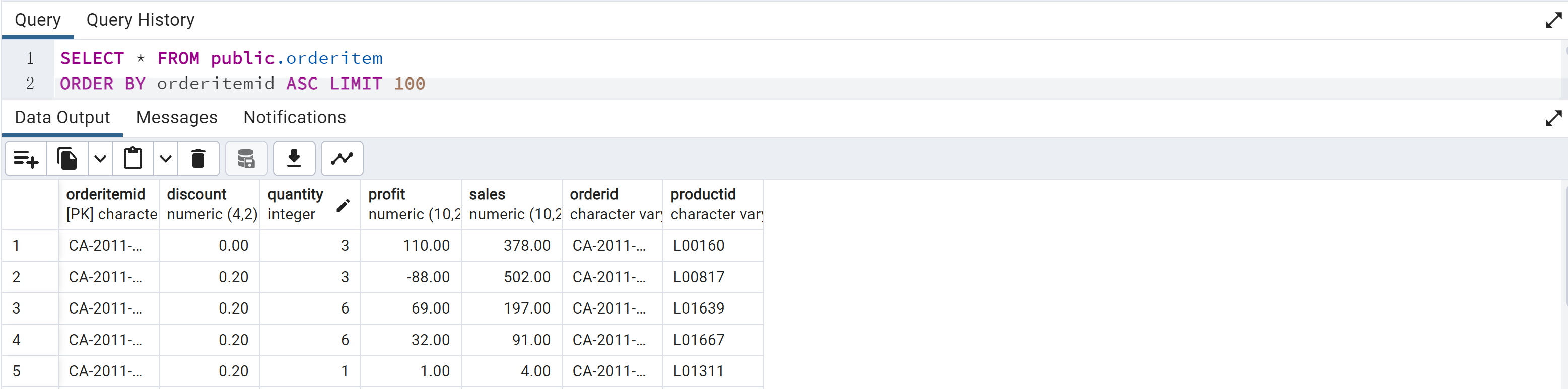
* **Product Table:**

****

* **Location Table:**

****

* **Order\_item Table:**

****

1. **3NF Normalization**

**Customer, Location, Order, and Order\_item relations are in 3NF**

**There is no partial functional and no transitive functional dependencies.**

* **Customer (CustomerID, CustomerName, Segment)**
* **FD1: CustomerID → CustomerName, Segment**
* **Location (PostalCode, City, State, Region, Country**
* **FD1: PostalCode → City, State, Region, Country**
* **Order (OrderID, OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK))**
* **FD1: OrderID → OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK)**
* **OrderItem (OrderItemID, Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK))**
* **FD1: OrderItemID → Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK)**

**Product is not in 3NF**

**Product is in 2NF since there is no partial functional dependencies, but there is transitive functional dependencies.**

* **Product (ProductID, ProductName, Manufacturer, SubCategory, Category**
* **FD1: ProductID → ProductName, Manufacturer, SubCategory, Category**
* **FD2: SubCategory → Category**

**Create a new relation for Category**

* **Category (SubCategory, Category)**
* **FD1: SubCategory → Category**
* **Product (ProductID, ProductName, Manufacturer, SubCategory)**
* **FD1: ProductID → ProductName, Manufacturer, SubCategory**

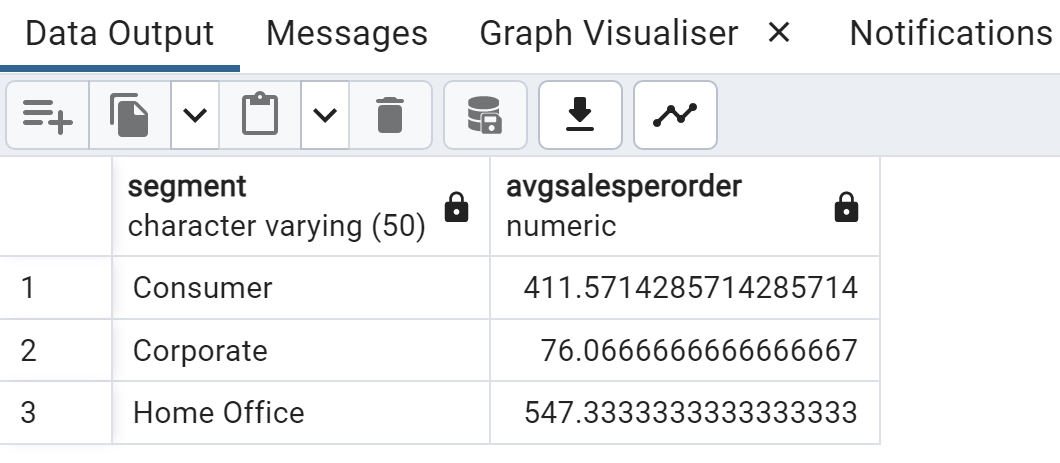
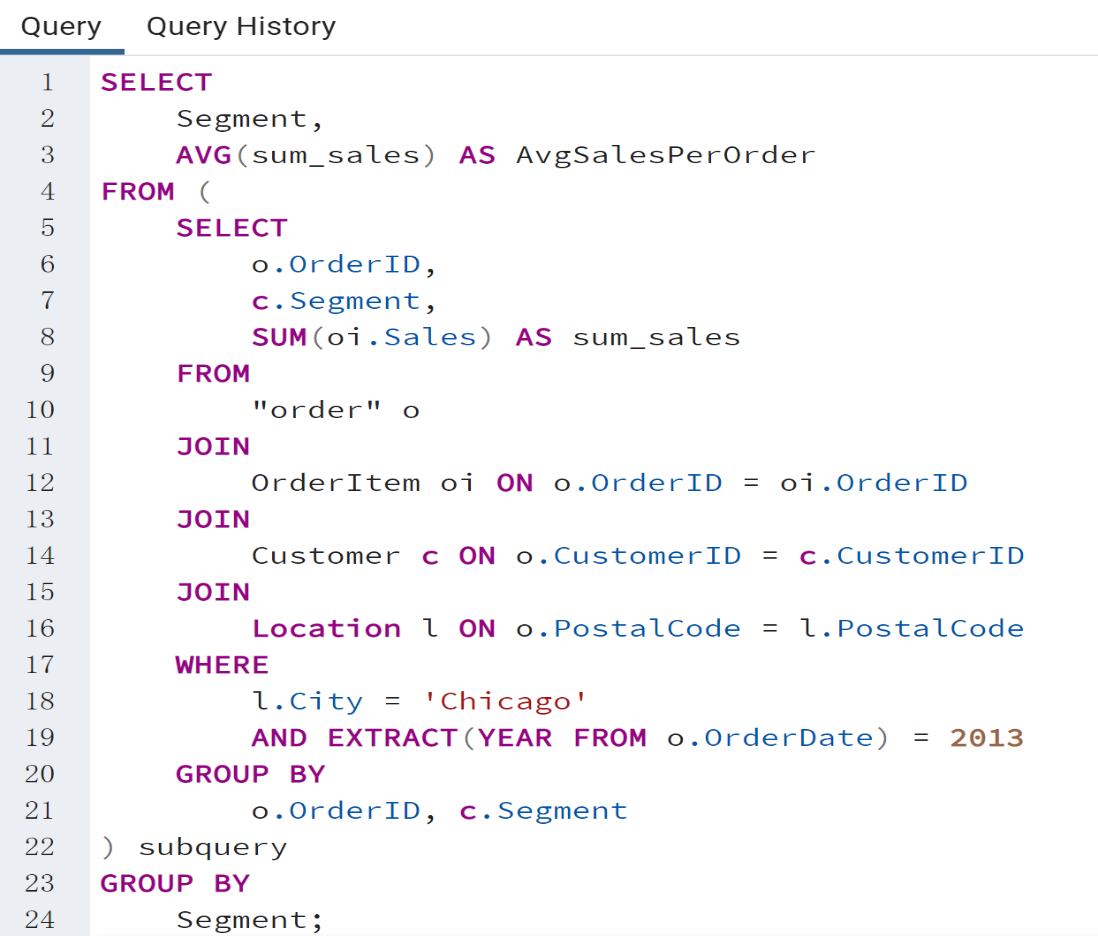
1. **Further Implementation**

**Final Relational Model is in 3NF**

* **Customer (CustomerID, CustomerName, Segment**
* **FD1: CustomerID → CustomerName, Segment)**
* **Location (PostalCode, City, State, Region, Country**
* **FD1: PostalCode → City, State, Region, Country**
* **Order (OrderID, OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK)**
* **FD1: OrderID → OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK)**
* **OrderItem (OrderItemID, Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK)**
* **FD1: OrderItemID → Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK)**
* **Category (SubCategory, Category)**
* **FD1: SubCategory → Category**
* **Product (ProductID, ProductName, Manufacturer, SubCategory)**
* **FD1: ProductID → ProductName, Manufacturer, SubCategory**

**Run a complex query in PostgreSQL**

* **Calculate the average sales per order in Chicago and 2013 by customer segment.**

****

**Compared to traditional Excel data storage, SQL offers several advantages:**

* **SQL's structure is more organized than Excel's, making it easier for users to understand.**
* **SQL querying is more adaptable than Excel, accommodating a diverse range of data retrieval needs.**
* **SQL has fewer data redundancies, leading to storage efficiency and reduced inconsistencies.**
* **For processing large volumes of data, SQL databases outperform Excel, delivering faster data retrieval times.**
* **The data integrity and protection features in SQL are more robust, ensuring superior security against potential threats.**