Project: Building A Relational Database using Docker, PostgreSQL and DBeaver

Tools Used

- Docker:

Docker (for windows) was downloaded and installed on the desktop for the purpose of containerizing (or housing) the PostgreSQL database to make it easier to manage and deploy the database

- **PostgreSQL:**

PosgreSQL was used because of its robust support for advanced and custom data types, data integrity and ACID compliance amongst other things. These features make it an excellent choice for enterprise level systems.

- **DBeaver:**

Dbeaver is an open-source tool and database management system that facilitates the development and management of the database via an interface. DBeaver was connected to the PostgreSQL container in Docker.

-Command Prompt

The CMD was used to execute the scripts, codes and commands using the **‘psql’** command line. The CMD was used for the installation of docker (desktop version), pulling the PostgreSQL image into Docker and containerising it; connecting DBeaver to the Postgresql container in docker and the development of the database.

Project Aim

Aim:

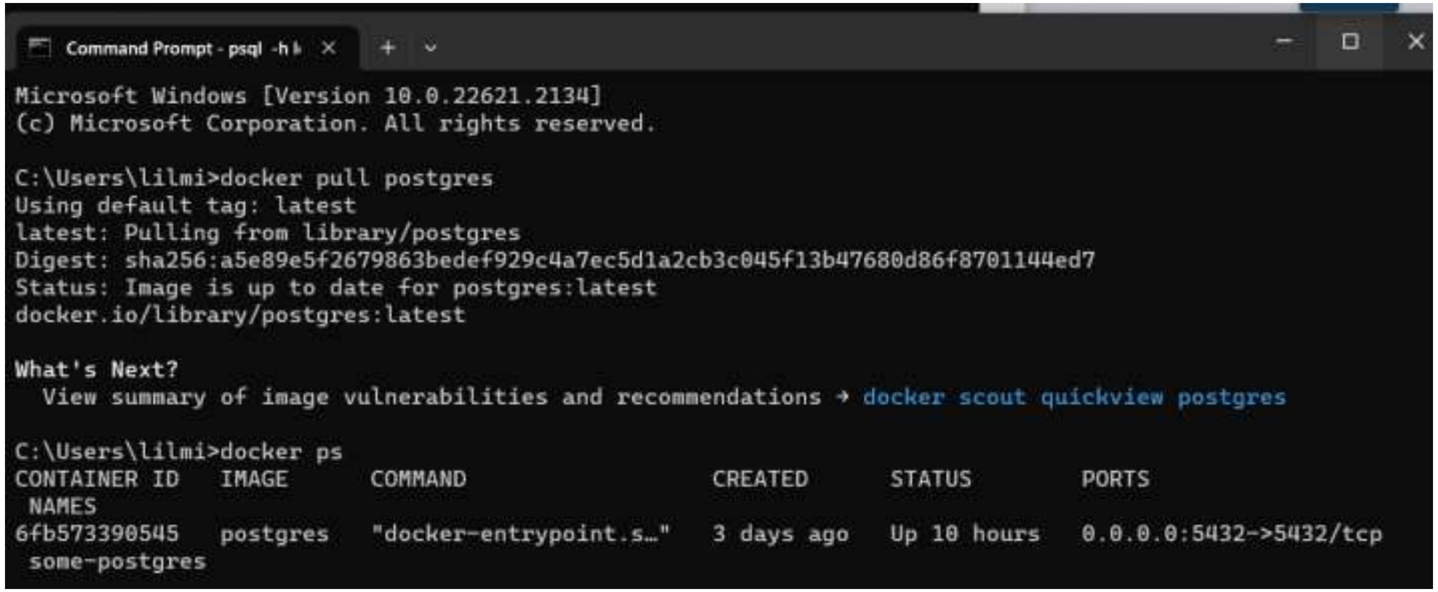
- Build a Retail Database named “retailDB” from scratch using PostgreSQL, Docker and DBeaver
- Employ the use of constraints, keys and referential integrity to ensure data consistency, quality and integrity in the creation of retailDB

Language used for creating retailDB:

- PSQL

Pulled the PostgreSQL image into Docker and created a container for it.

Container name: “some-postgres”

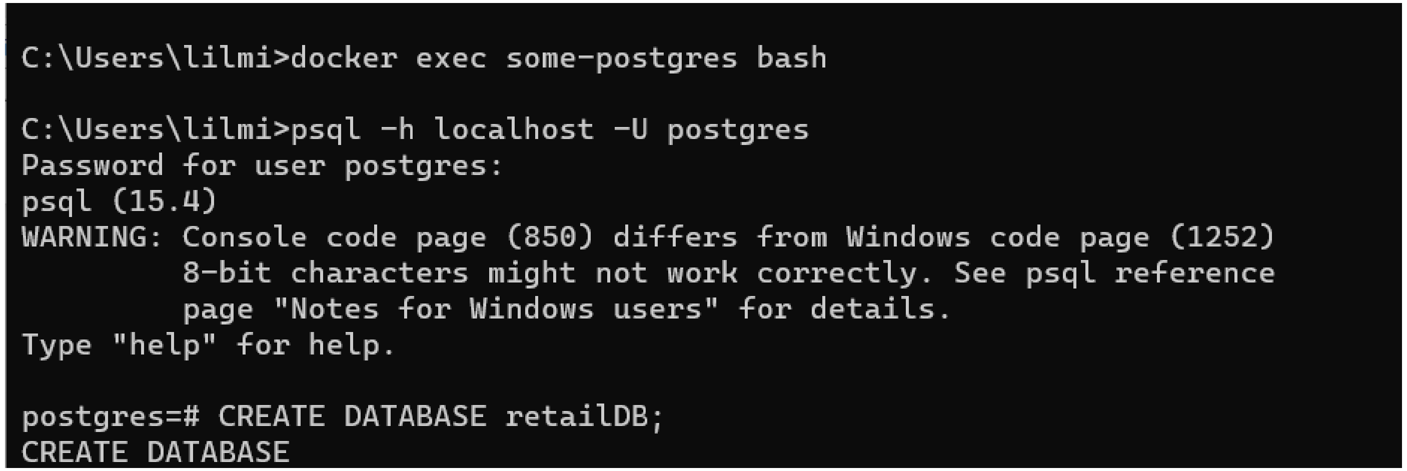


```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lilmi>docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
Digest: sha256:a5e89e5f2679863bedef929c4a7ec5d1a2cb3c045f13b47680d86f8701144ed7
Status: Image is up to date for postgres:latest
docker.io/library/postgres:latest

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview postgres

C:\Users\lilmi>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED       STATUS        PORTS
 NAMES
6fb573390545   postgres   "docker-entrypoint.s…"   3 days ago    Up 10 hours   0.0.0.0:5432->5432/tcp
 some-postgres
```

Using the windows command prompt, I created a database in the “some-postgres” container.

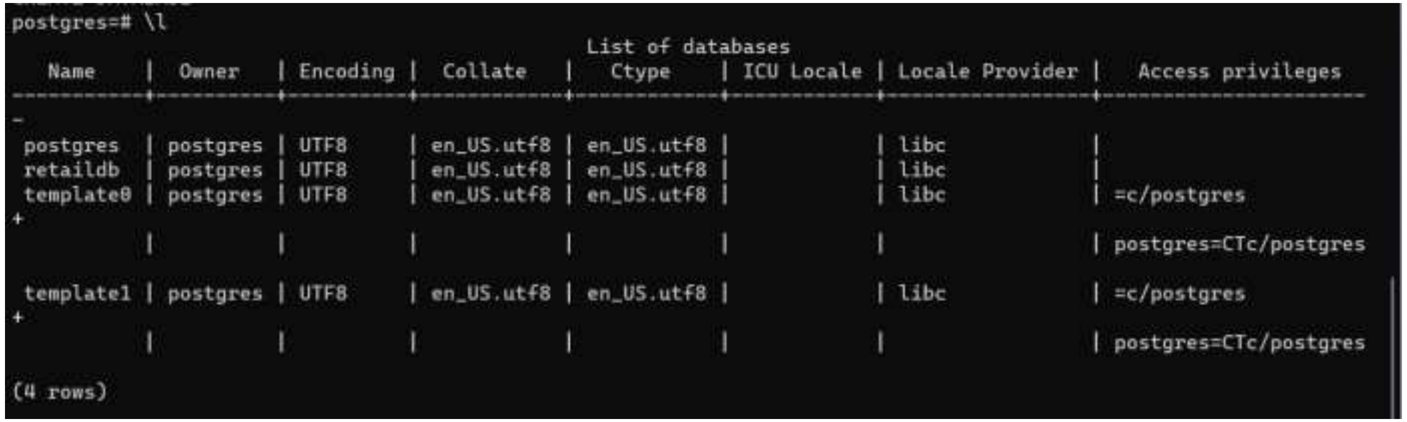
Database name= “retailDB”

User = “postgres”

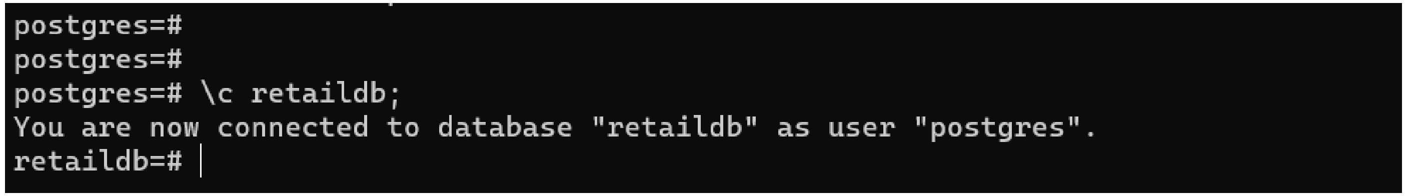


```
C:\Users\lilmi>docker exec some-postgres bash

C:\Users\lilmi>psql -h localhost -U postgres
Password for user postgres:
psql (15.4)
WARNING: Console code page (850) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# CREATE DATABASE retailDB;
CREATE DATABASE
```

List of databases created by user ‘postgres’ shown below:

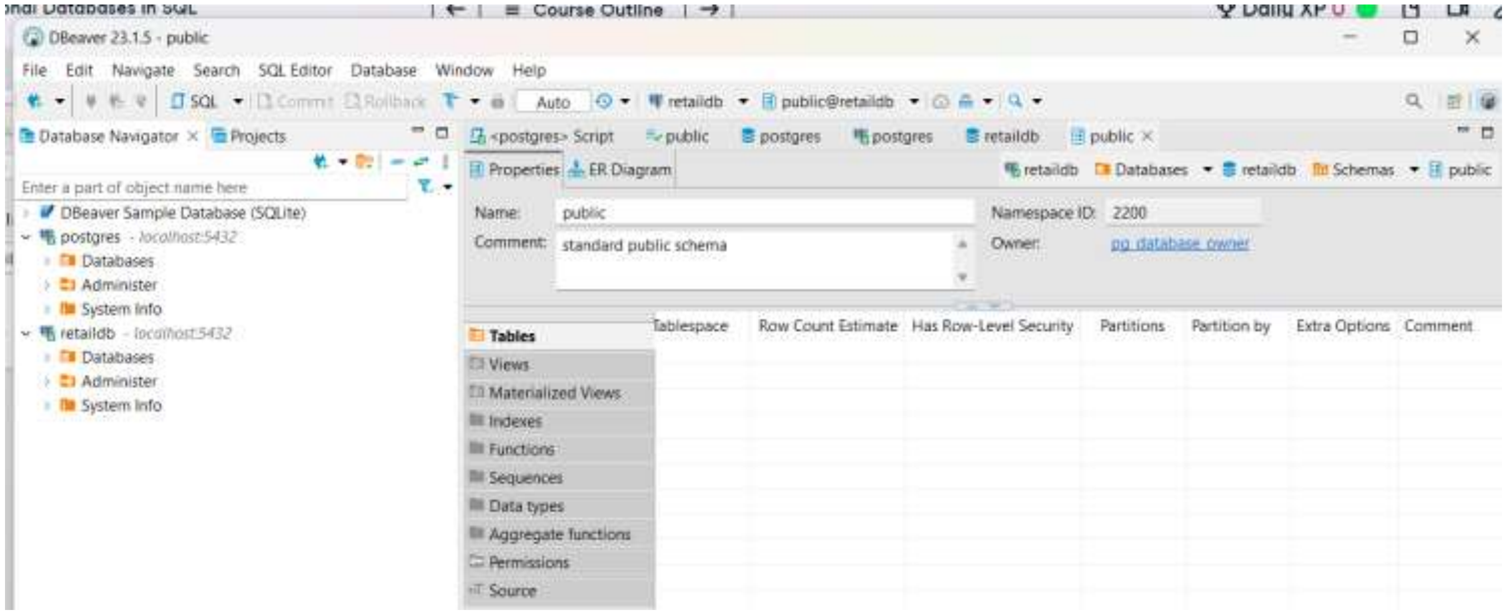


```
postgres=# \l
                                                  List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    | ICU Locale | Locale Provider |   Access privileges
-----------+----------+----------+------------+------------+------------+-----------------+-----------------------
-
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            |
 retaildb  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            | =c/postgres
+
           |          |          |            |            |            |                 | postgres=CTc/postgres

 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            | =c/postgres
+
           |          |          |            |            |            |                 | postgres=CTc/postgres

(4 rows)
```



```
postgres=#
postgres=#
postgres=# \c retaildb;
You are now connected to database "retaildb" as user "postgres".
retaildb=#
```

Snapshot of the newly created “retailDB” database in DBeaver.



Name: public

Comment: standard public schema

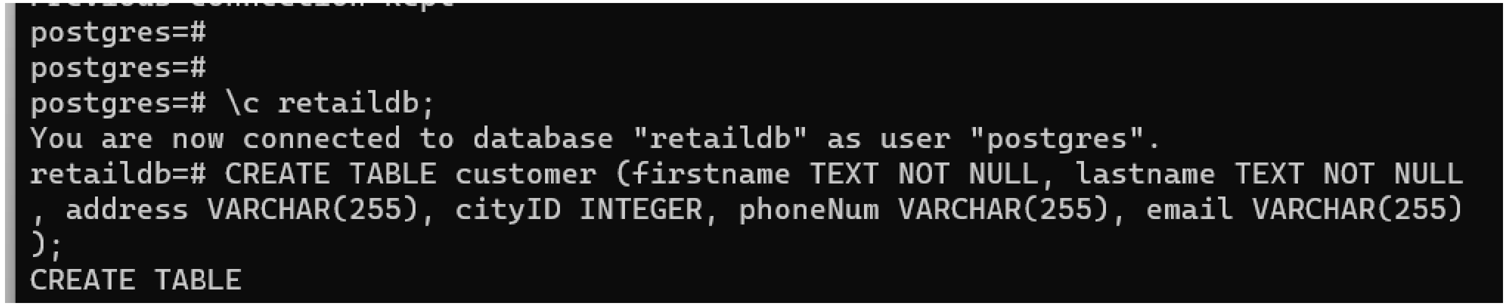
Namespace ID: 2200

Owner: pg_database_owner

Created the first table in the database called "customer"

Then altered the table to add a primary key constraint using the "Serial" keyword.

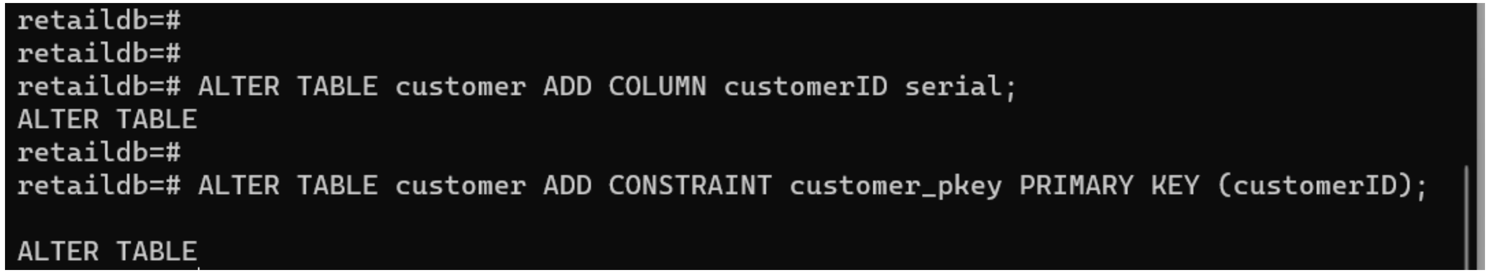
- Created the first table 'customer' in the retailDB database.



```
postgres=#
postgres=#
postgres=# \c retaildb;
You are now connected to database "retaildb" as user "postgres".
retaildb=# CREATE TABLE customer (firstname TEXT NOT NULL, lastname TEXT NOT NULL
, address VARCHAR(255), cityID INTEGER, phoneNum VARCHAR(255), email VARCHAR(255)
);
CREATE TABLE
```

- Altered the table to add a Primary Key constraint called "customerID"

Altered the table 'customer' to add a Primary Key constraint with the auto-increment feature specified by the 'SERIAL' keyword, so that each new customer record that is added is automatically given a unique 'customerID' that uniquely identifies each row of the customer table.



```
retaildb=#
retaildb=#
retaildb=# ALTER TABLE customer ADD COLUMN customerID serial;
ALTER TABLE
retaildb=#
retaildb=# ALTER TABLE customer ADD CONSTRAINT customer_pkey PRIMARY KEY (customerID);

ALTER TABLE
```

Snapshot of the newly created table in DBeaver

- Customer column with the Primary Key column “customerID”

The columns of the customer table are displayed with their data types. The columns with the ‘NOT NULL’ key constraint are signified by ‘v’. The NOT NULL columns are firstname, lastname and customerid meaning they cannot accept null values.

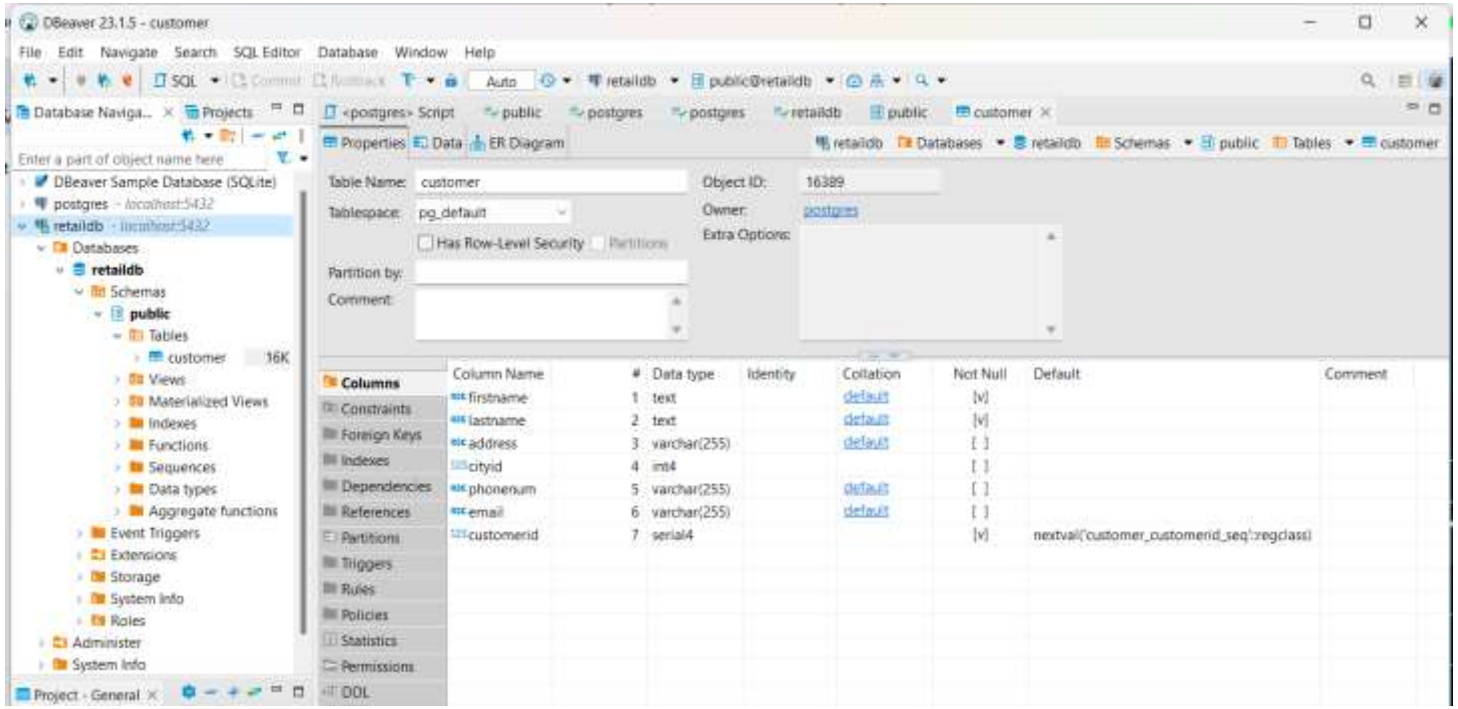


Table Name: customer

Tablespace: pg_default

☐ Has Row-Level Security ☐ Partitions

Partition by:

Comment:

Object ID: 16389

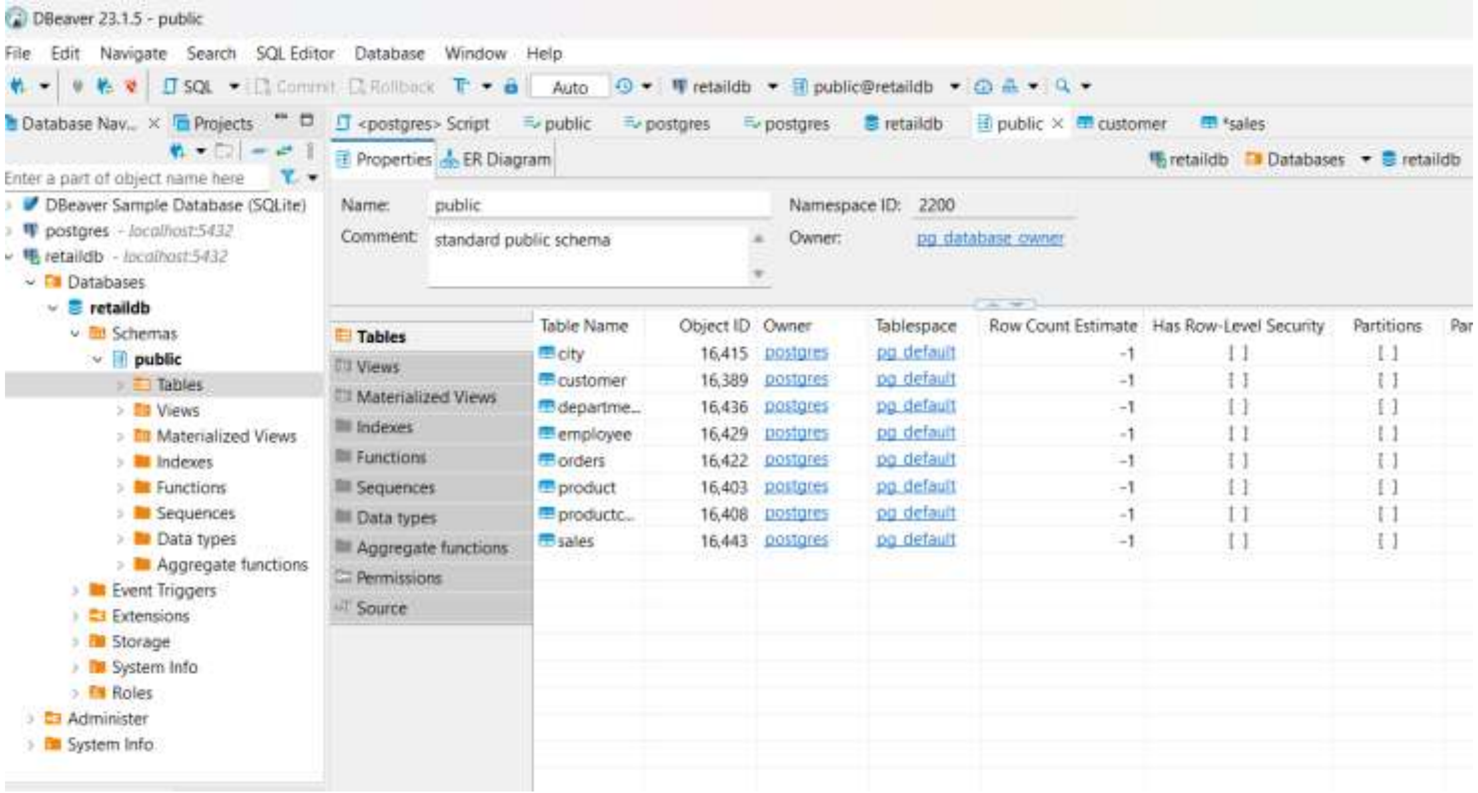
Owner: postgres

Extra Options:

Column Name	#	Data type	Identity	Collation	Not Null	Default	Comment
firstname	1	text		default	[v]		
lastname	2	text		default	[v]		
address	3	varchar(255)		default	[ ]		
cityid	4	int4			[ ]		
phonenum	5	varchar(255)		default	[ ]		
email	6	varchar(255)		default	[ ]		
customerid	7	serial4			[v]	nextval('customer_customerid_seq'::regclass)	

Snapshot showing the creation of 7 more tables (in addition to the customer table) including the fact table “Sales”

- Creation of 7 new tables using psql in the command prompt
  - ➢ 7 Dimension tables: customer (already created), city, department, employee, orders, product and product category.
  - ➢ 1 Fact Table: Sales



Name: public

Namespace ID: 2200

Comment: standard public schema

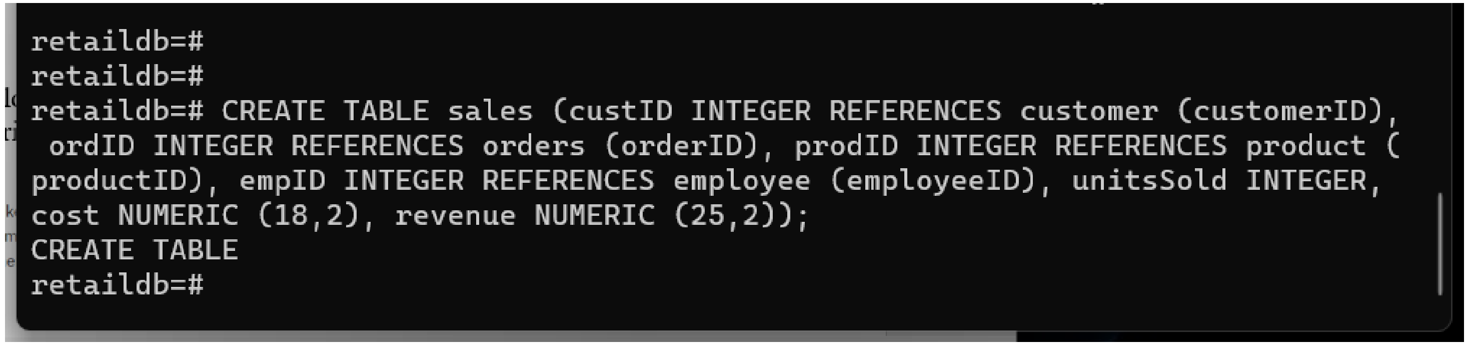
Owner: pg_database_owner

Table Name	Object ID	Owner	Tablespace	Row Count Estimate	Has Row-Level Security	Partitions	Par
city	16,415	postgres	pg_default	-1	☐	☐	
customer	16,389	postgres	pg_default	-1	☐	☐	
departme...	16,436	postgres	pg_default	-1	☐	☐	
employee	16,429	postgres	pg_default	-1	☐	☐	
orders	16,422	postgres	pg_default	-1	☐	☐	
product	16,403	postgres	pg_default	-1	☐	☐	
productc...	16,408	postgres	pg_default	-1	☐	☐	
sales	16,443	postgres	pg_default	-1	☐	☐	

Creation of the Fact table “Sales” in Command Prompt using psql.

- Facts of the Facts table

- ➢ The primary key for the Sales table is a combination of the foreign keys being referenced from certain dimension tables (customer, orders, products and employee)



```
retaildb=#
retaildb=#
retaildb=# CREATE TABLE sales (custID INTEGER REFERENCES customer (customerID),
 ordID INTEGER REFERENCES orders (orderID), prodID INTEGER REFERENCES product (
productID), empID INTEGER REFERENCES employee (employeeID), unitsSold INTEGER,
cost NUMERIC (18,2), revenue NUMERIC (25,2));
CREATE TABLE
retaildb=#
```

- Columns of the Sales table in DBeaver

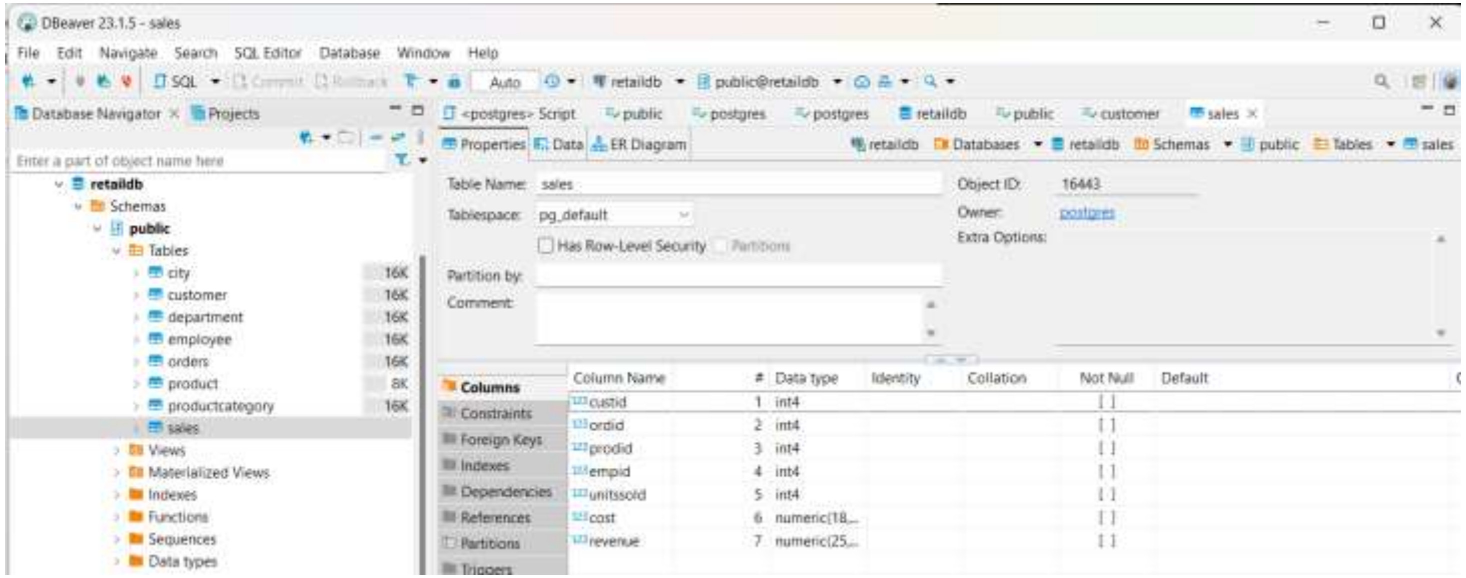


Table Name: sales

Tablespace: pg_default

□ Has Row-Level Security □ Partitions

Partition by:

Comment:

Object ID: 16443

Owner: postgres

Extra Options:

Column Name	#	Data type	Identity	Collation	Not Null	Default
custid	1	int4			[ ]	
ordid	2	int4			[ ]	
prodid	3	int4			[ ]	
empid	4	int4			[ ]	
unitssold	5	int4			[ ]	
cost	6	numeric(18,...			[ ]	
revenue	7	numeric(25,...			[ ]	

Enforcing Referential Integrity on the columns of the Sales table.

Sales Table showing the Foreign Key columns & Referenced Columns & Associated Tables

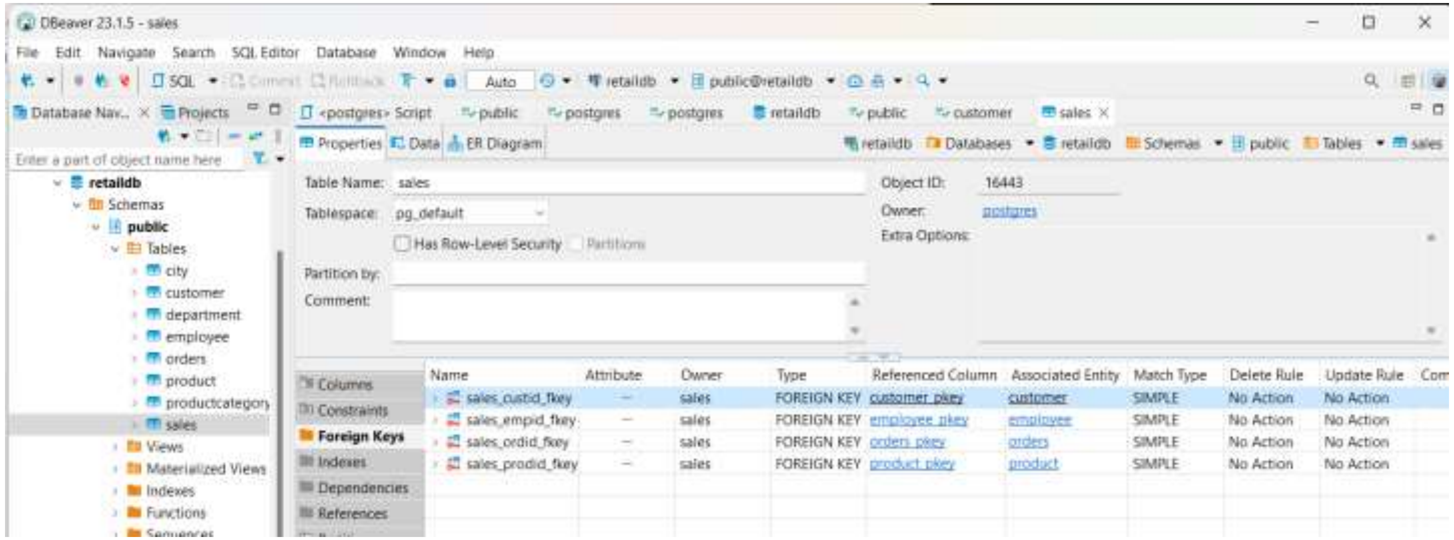


Table Name: sales

Tablespace: pg_default

□ Has Row-Level Security □ Partitions

Partition by:

Comment:

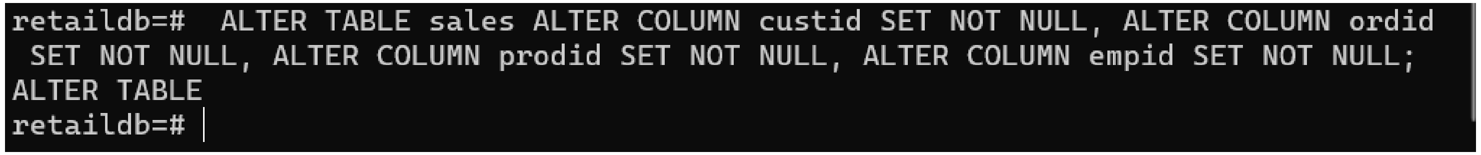
Object ID: 16443

Owner: postgres

Extra Options:

Name	Attribute	Owner	Type	Referenced Column	Associated Entity	Match Type	Delete Rule	Update Rule	Com
sales_custid_fkey	—	sales	FOREIGN KEY	customer_pkey	customer	SIMPLE	No Action	No Action	
sales_empid_fkey	—	sales	FOREIGN KEY	employee_pkey	employee	SIMPLE	No Action	No Action	
sales_ordid_fkey	—	sales	FOREIGN KEY	orders_pkey	orders	SIMPLE	No Action	No Action	
sales_prodid_fkey	—	sales	FOREIGN KEY	product_pkey	product	SIMPLE	No Action	No Action	

However, foreign key columns by default accept duplicates and null values even if they reference a primary key column in another table. **The problem here does not lie with the duplicate values but rather, the null values,** if null values are allowed in the foreign key columns of the sales table, then it may cause inconsistences in data and ultimately pose a problem for analytics. **Hence, the solution is to set a not null constraint in the foreign key columns of the Sales table using command prompt as shown below;**



```
retaildb=#  ALTER TABLE sales ALTER COLUMN custid SET NOT NULL, ALTER COLUMN ordid
 SET NOT NULL, ALTER COLUMN prodid SET NOT NULL, ALTER COLUMN empid SET NOT NULL;
ALTER TABLE
retaildb=# |
```

Enforcing Referential Integrity on the columns of the Sales table by:

1; Adding the ‘NOT NULL’ constraint to the foreign key columns. This will facilitate a quality relationship between the sales table and the dimension tables thereby promoting data consistency.

NOT NULL constraint on the foreign key columns denoted by ‘V’

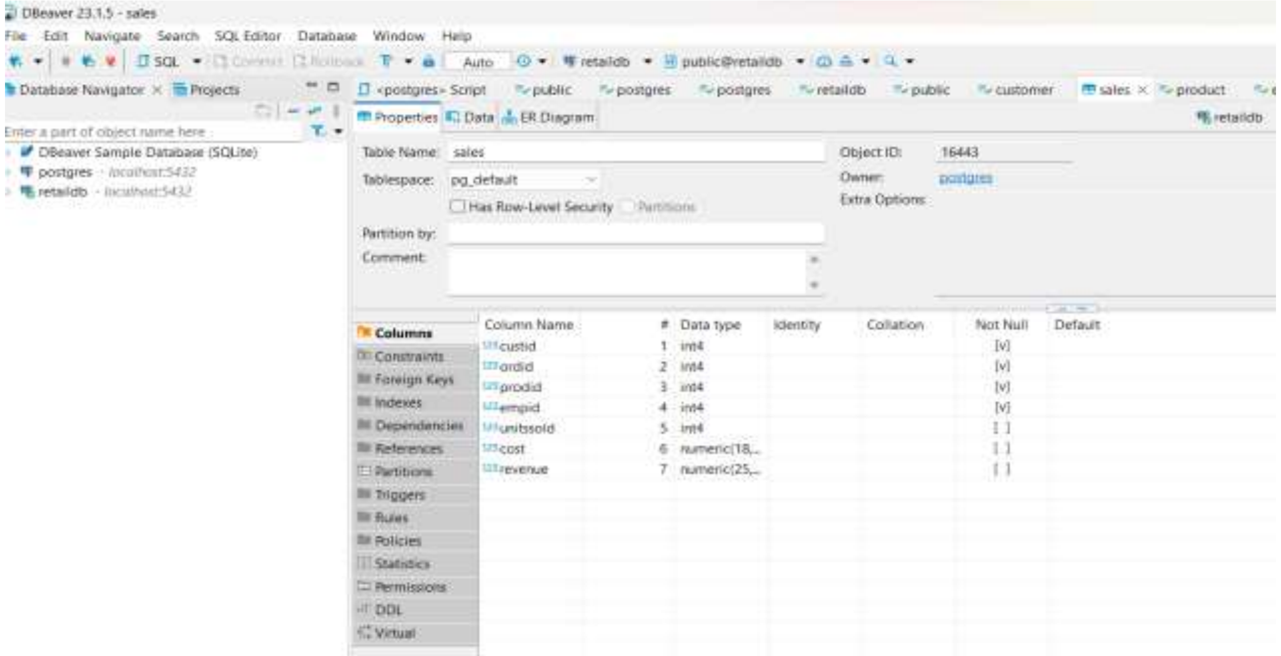


Table Name: sales
Tablespace: pg_default
Object ID: 16443
Owner: postgres

Column Name	#	Data type	Identity	Collation	Not Null	Default
custid	1	int4			[v]	
ordid	2	int4			[v]	
prodid	3	int4			[v]	
empid	4	int4			[v]	
unitssold	5	int4			[ ]	
cost	6	numeric(18,...			[ ]	
revenue	7	numeric(25,...			[ ]	

PSQL code showing constraints on the Sales Table

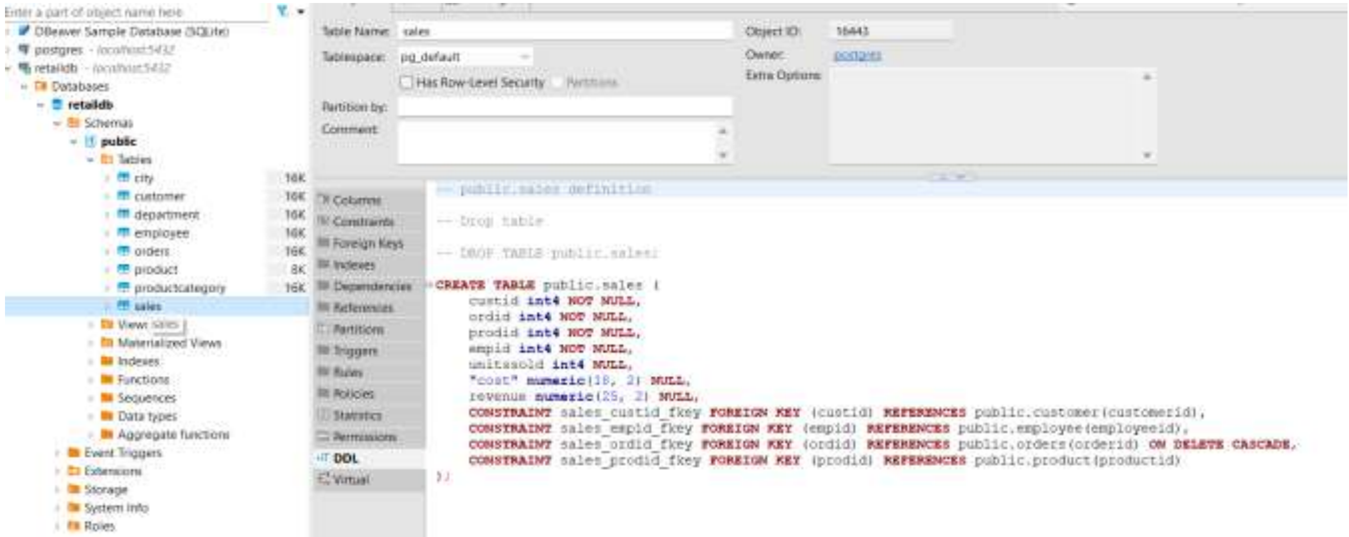


Table Name: sales
Tablespace: pg_default
Object ID: 16443
Owner: postgres

```
-- public.sales definition

-- Drop table

-- DROP TABLE public.sales;

CREATE TABLE public.sales (
	custid int4 NOT NULL,
	ordid int4 NOT NULL,
	prodid int4 NOT NULL,
	empid int4 NOT NULL,
	unitssold int4 NULL,
	"cost" numeric(18, 2) NULL,
	revenue numeric(25, 2) NULL,
	CONSTRAINT sales_custid_fkey FOREIGN KEY (custid) REFERENCES public.customer(customerid),
	CONSTRAINT sales_empid_fkey FOREIGN KEY (empid) REFERENCES public.employee(employeeid),
	CONSTRAINT sales_ordid_fkey FOREIGN KEY (ordid) REFERENCES public.orders(orderid) ON DELETE CASCADE,
	CONSTRAINT sales_prodid_fkey FOREIGN KEY (prodid) REFERENCES public.product(productid)
);
```

Enforcing Referential Integrity on the columns of the Sales table by

2; Specify the actions to be taken on the Facts table (sales) if a record is deleted from one of the referenced dimensions table.

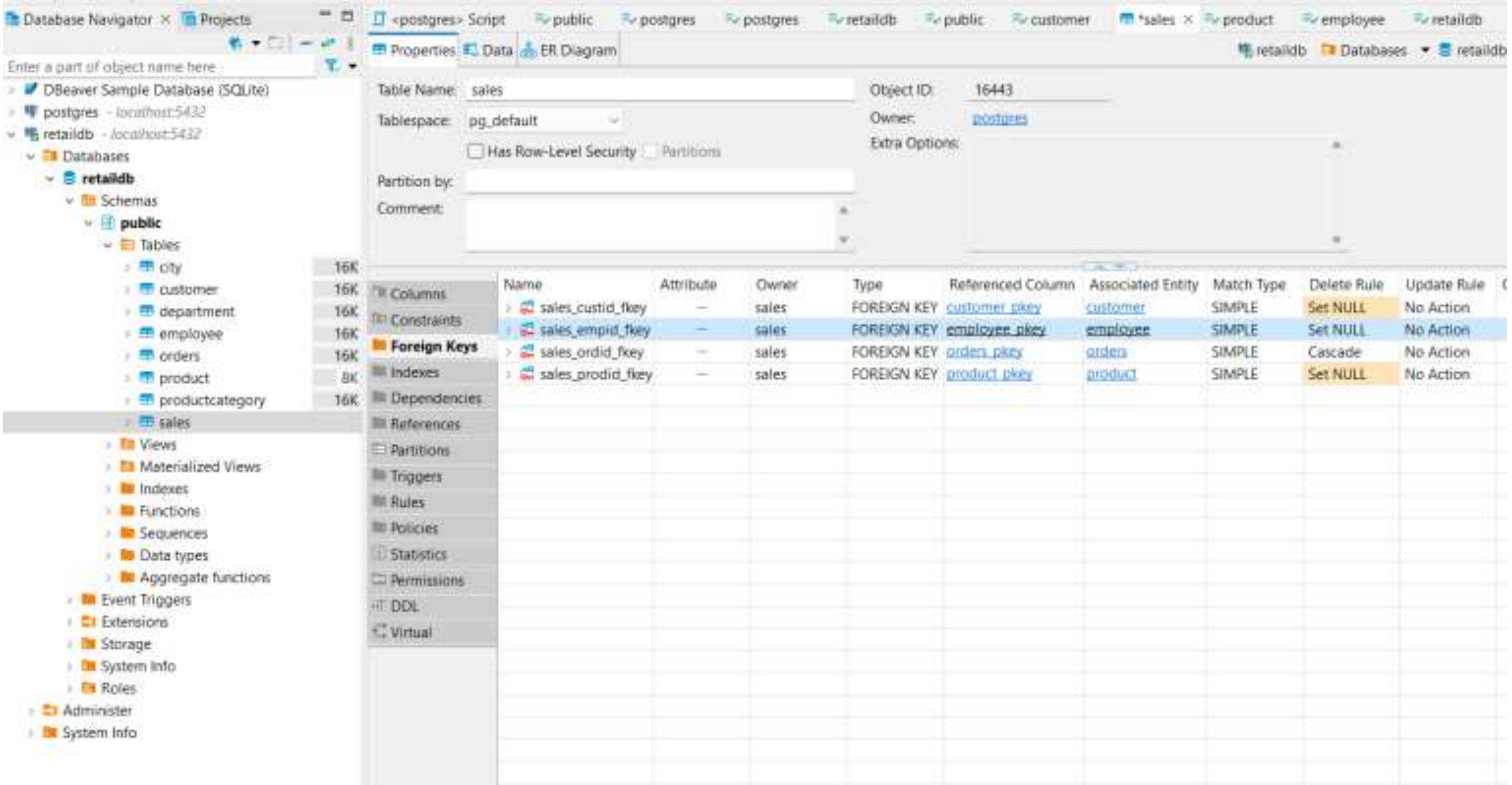


Table Name: sales

Tablespace: pg_default

☐ Has Row-Level Security ☐ Partitions

Partition by:

Comment:

Object ID: 16443

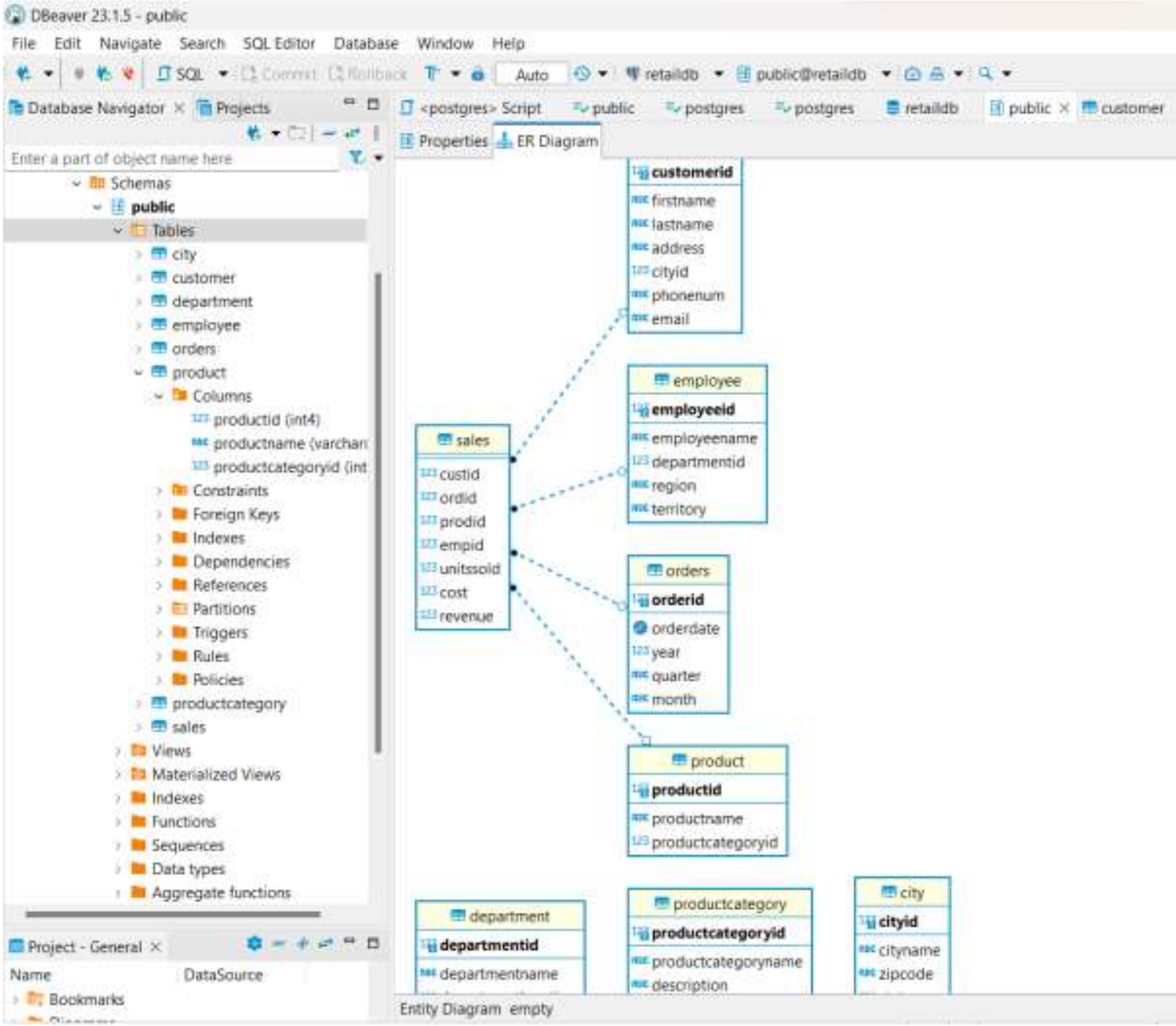
Owner: postgres

Extra Options:

Name	Attribute	Owner	Type	Referenced Column	Associated Entity	Match Type	Delete Rule	Update Rule
sales_custid_fkey	–	sales	FOREIGN KEY	customer_pkey	customer	SIMPLE	Set NULL	No Action
sales_empid_fkey	–	sales	FOREIGN KEY	employee_pkey	employee	SIMPLE	Set NULL	No Action
sales_ordid_fkey	–	sales	FOREIGN KEY	orders_pkey	orders	SIMPLE	Cascade	No Action
sales_prodid_fkey	–	sales	FOREIGN KEY	product_pkey	product	SIMPLE	Set NULL	No Action

Following the diagram, if a record is deleted from the orders table, then the associated record should be automatically deleted from sales table. For all the other dimension tables, the 'delete rule' is 'Set NULL meaning if a record is deleted in the associated tables, then it should appear as null in the sales table which may lead to some sort of investigation by whoever is handling the data.

Current Entity Relationship Diagram



customer: customerid, firstname, lastname, address, cityid, phonenum, email

employee: employeeid, employeename, departmentid, region, territory

sales: custid, ordid, prodid, empid, unitssold, cost, revenue

orders: orderid, orderdate, year, quarter, month

product: productid, productname, productcategoryid

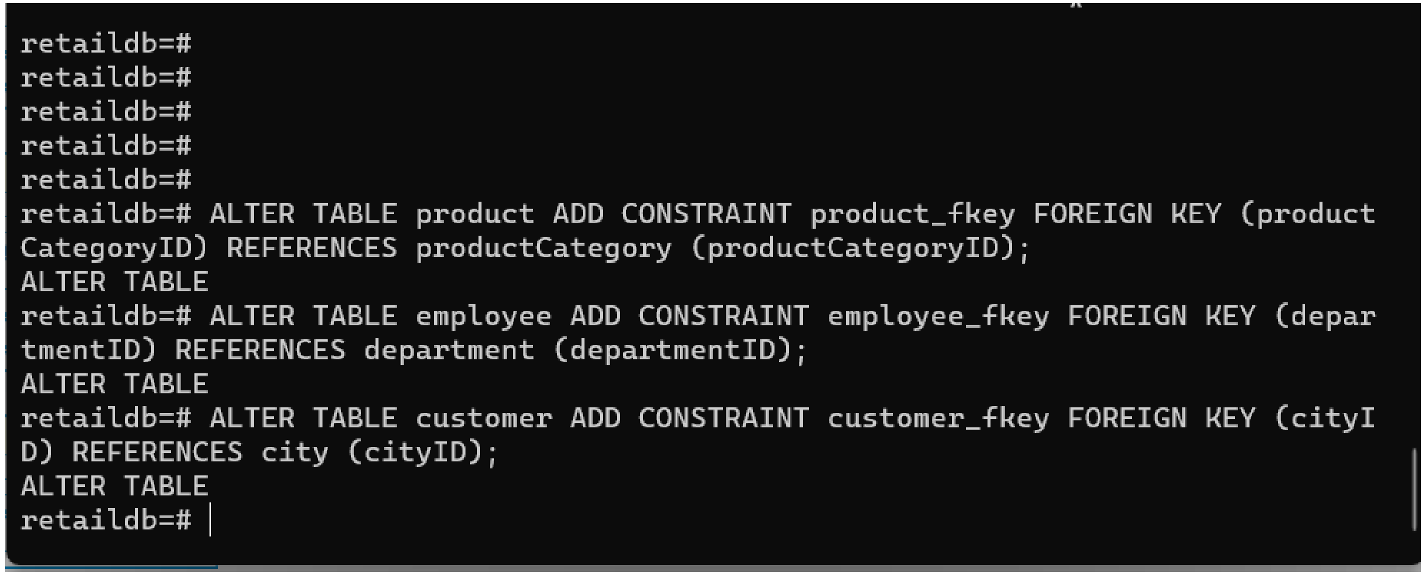
department: departmentid, departmentname

productcategory: productcategoryid, productcategoryname, description

city: cityid, cityname, zipcode

From the diagram, the fact table is connected to only 4 dimension tables which is not the model we're looking for. Hence, a relationship must be established between the facts table and the remaining dimensions table to achieve a 'star schema' or a 'snowflake schema'.

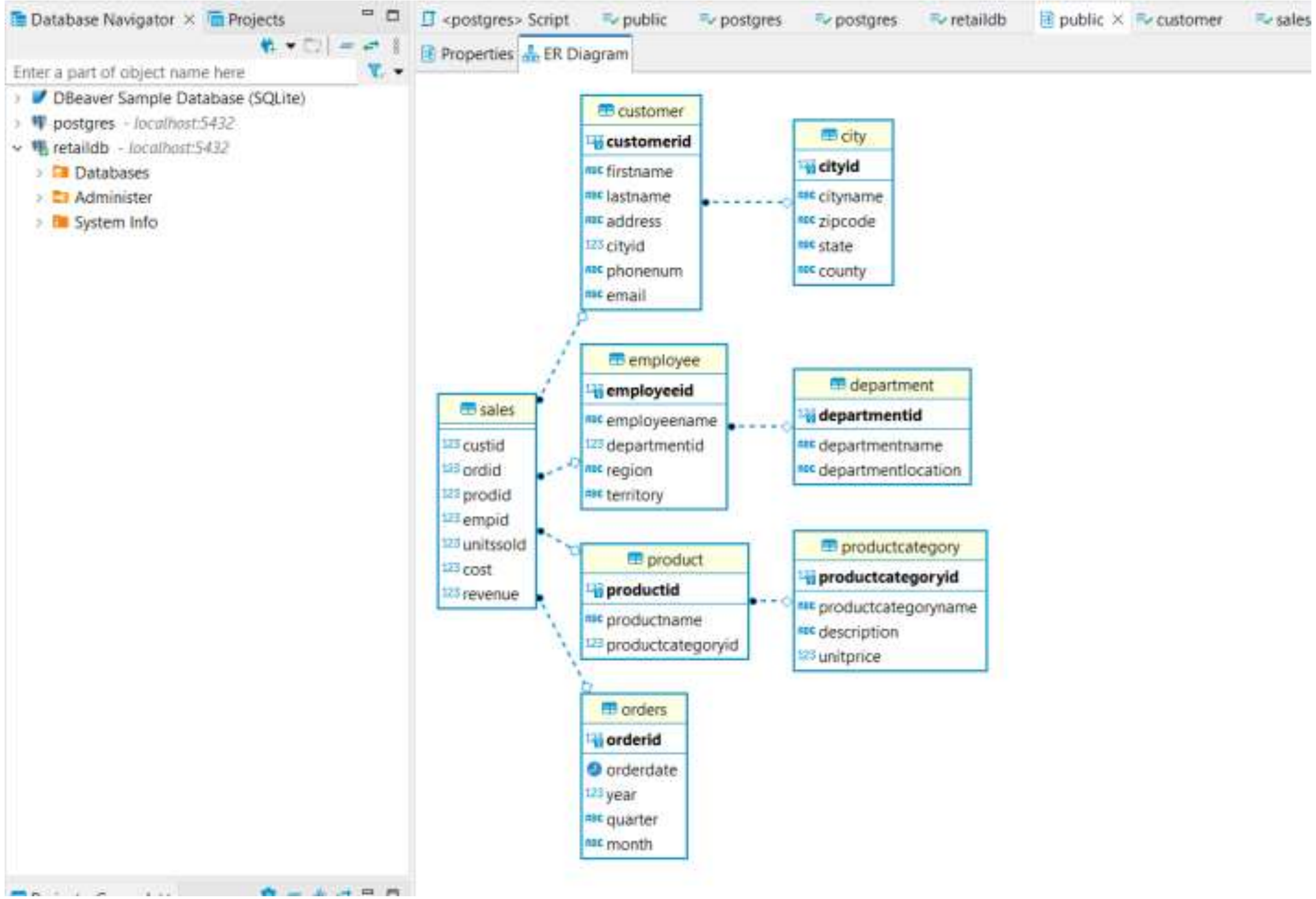
Establishing relationships between the tables by using the 'REFERENCE' Keyword



```
retaildb=#
retaildb=#
retaildb=#
retaildb=#
retaildb=#
retaildb=# ALTER TABLE product ADD CONSTRAINT product_fkey FOREIGN KEY (product
CategoryID) REFERENCES productCategory (productCategoryID);
ALTER TABLE
retaildb=# ALTER TABLE employee ADD CONSTRAINT employee_fkey FOREIGN KEY (depar
tmentID) REFERENCES department (departmentID);
ALTER TABLE
retaildb=# ALTER TABLE customer ADD CONSTRAINT customer_fkey FOREIGN KEY (cityI
D) REFERENCES city (cityID);
ALTER TABLE
retaildb=#
```

Remodelled Entity Relationship Diagram

Remodelled Entity Relationship Diagram with a “**Snowflake Schema**”


Database Navigator
Projects
Enter a part of object name here
DBeaver Sample Database (SQLite)
postgres - localhost:5432
retaildb - localhost:5432
Databases
Administer
System Info
<postgres> Script
public
postgres
postgres
retaildb
public
customer
sales
Properties
ER Diagram
customer
customerid
firstname
lastname
address
cityid
phonenum
email
city
cityid
cityname
zipcode
state
county
employee
employeeid
employeename
departmentid
region
territory
department
departmentid
departmentname
departmentlocation
sales
custid
ordid
prodid
empid
unitssold
cost
revenue
product
productid
productname
productcategoryid
productcategory
productcategoryid
productcategoryname
description
unitprice
orders
orderid
orderdate
year
quarter
month