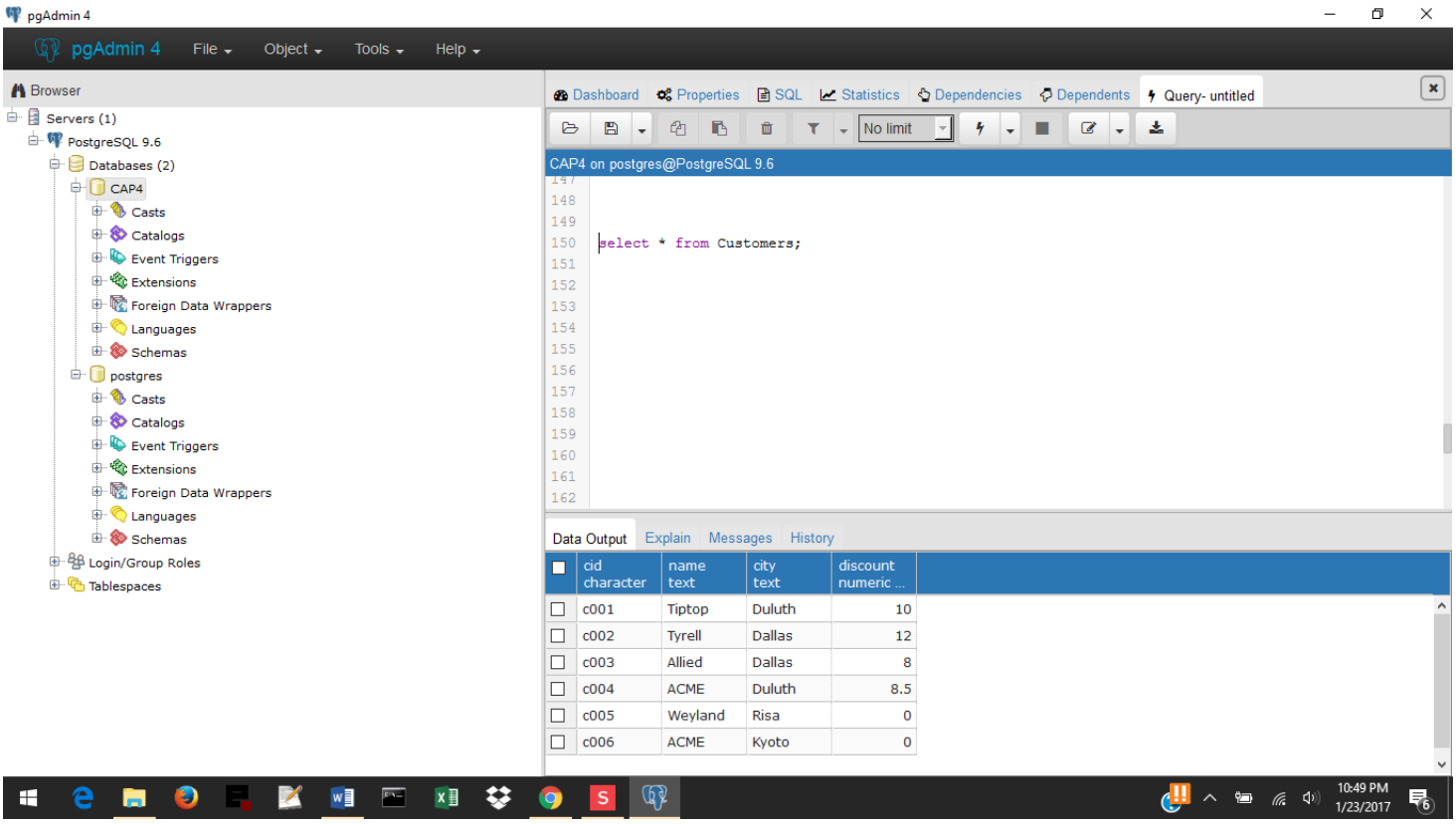
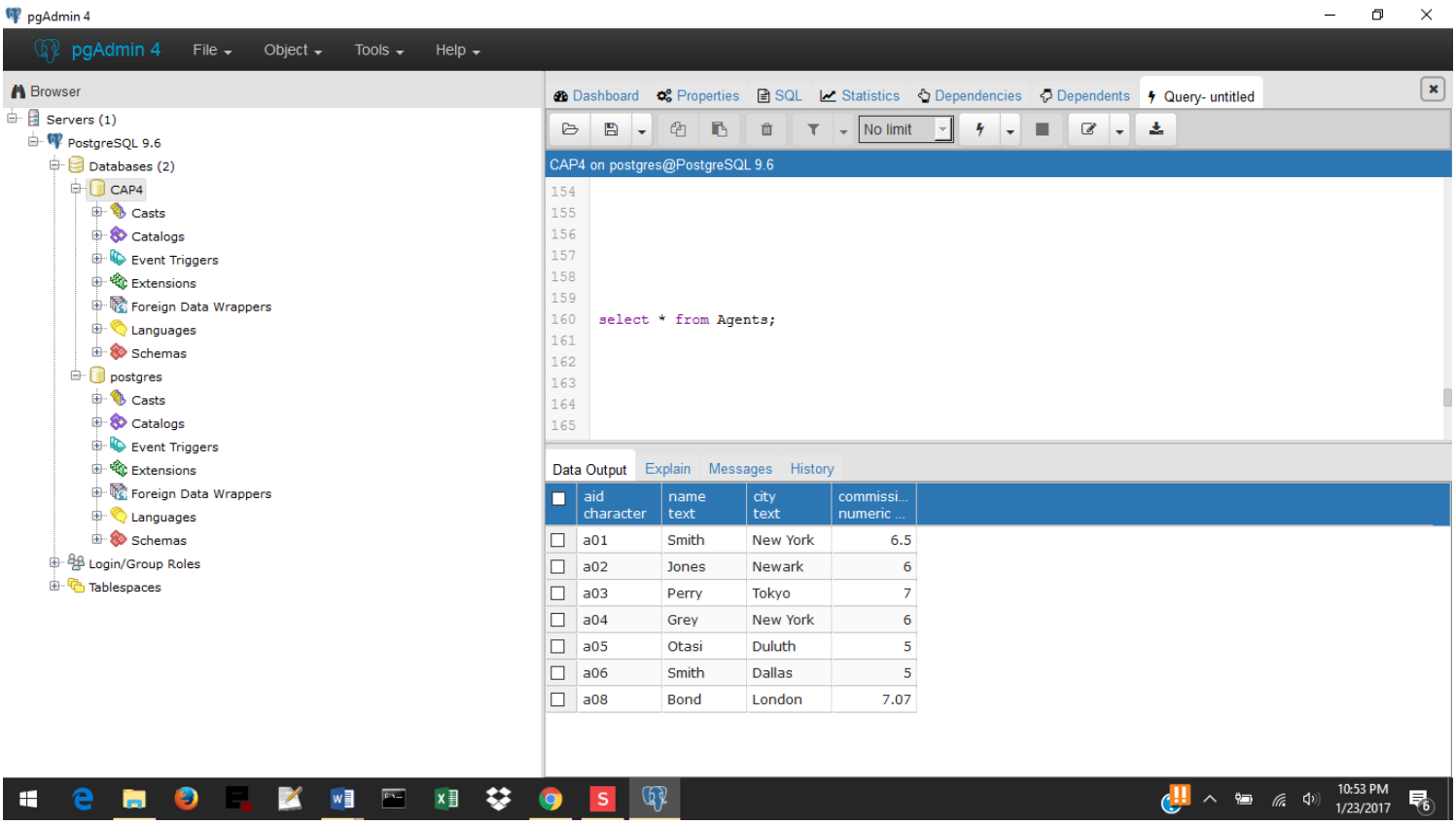
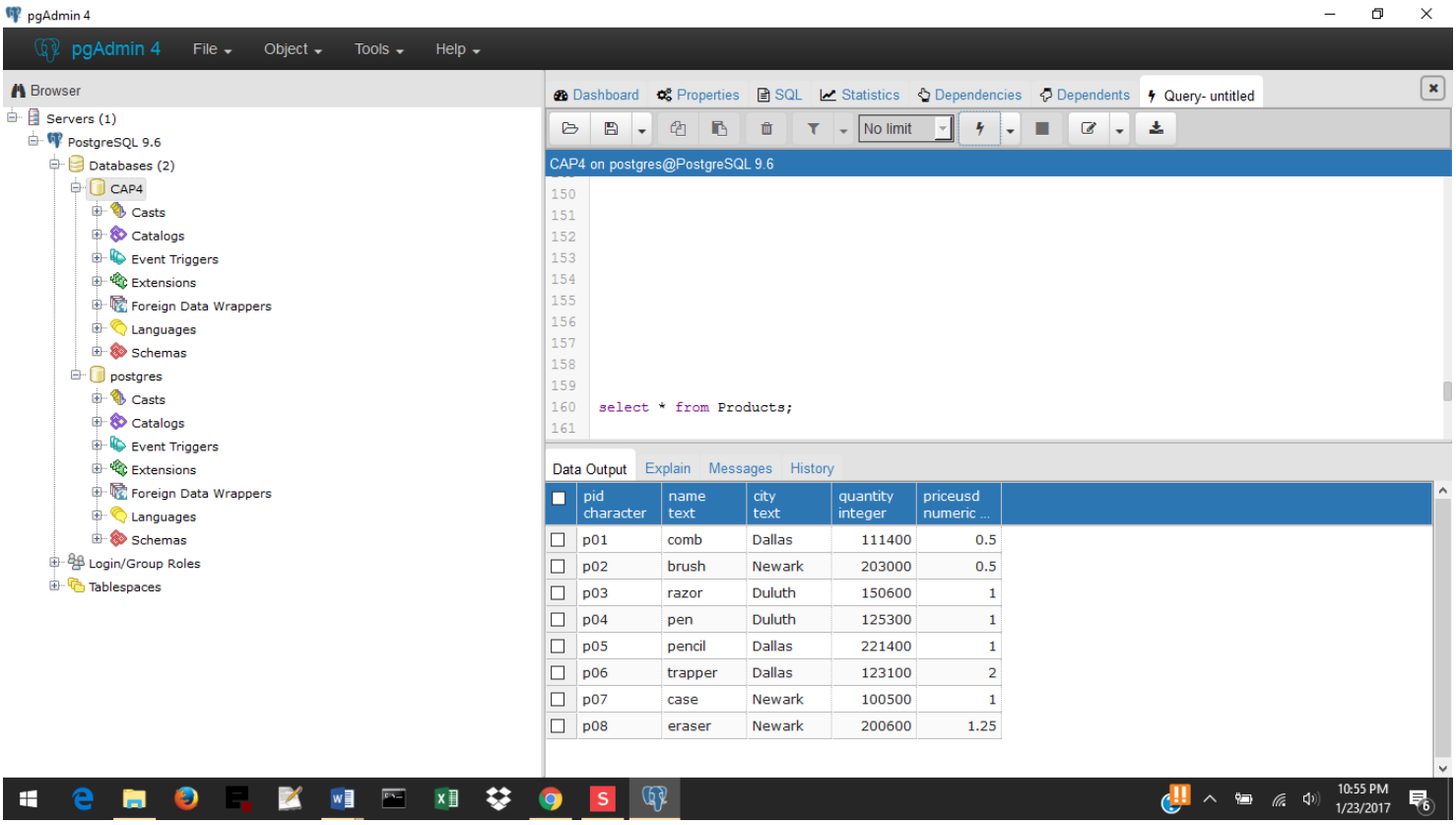
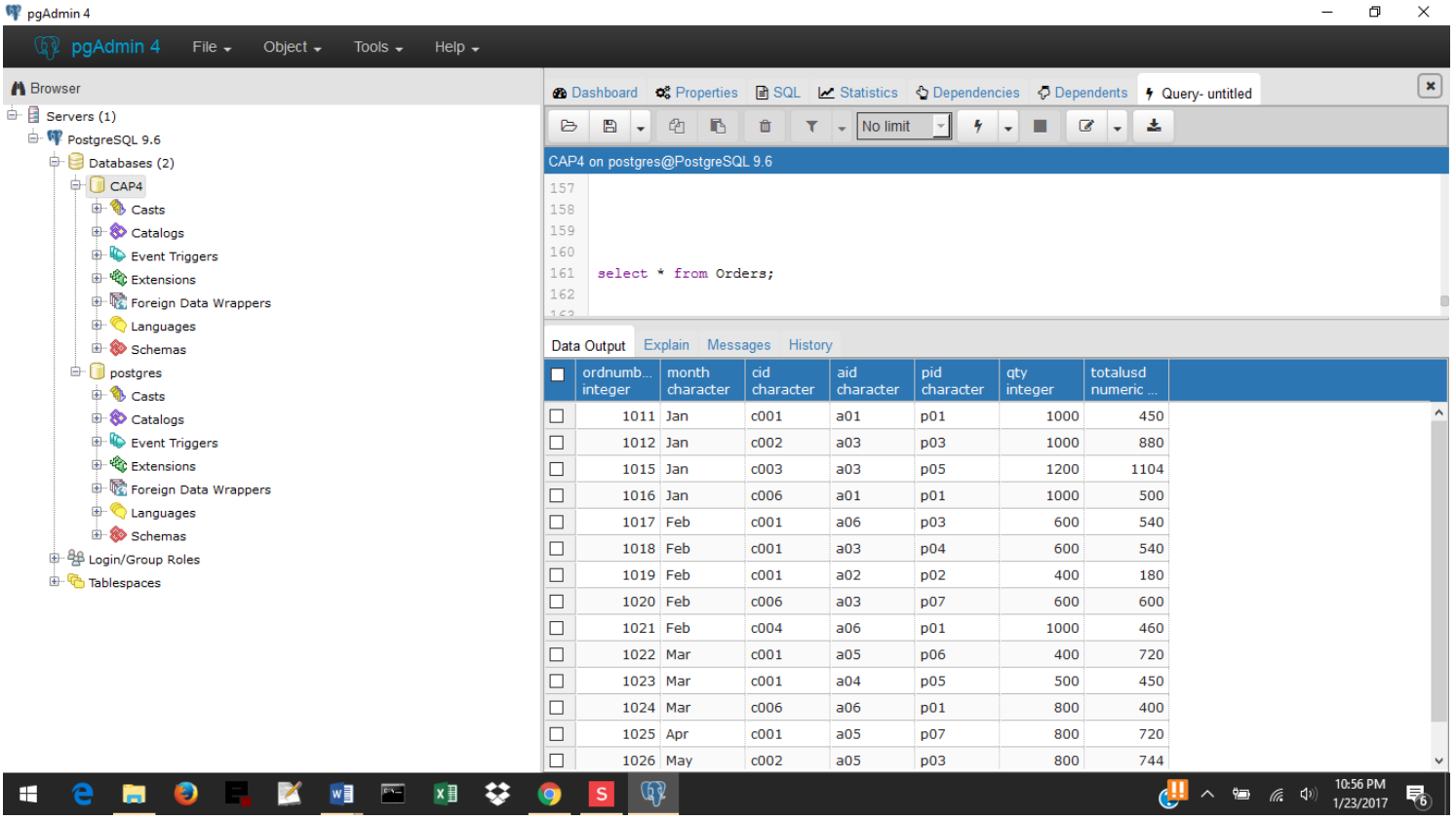

pgAdmin 4
File
Object
Tools
Help
Browser
Servers (1)
PostgreSQL 9.6
Databases (2)
CAP4
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Query- untitled
No limit
CAP4 on postgres@PostgreSQL 9.6
select * from Customers;
Data Output
Explain
Messages
History
cid character
name text
city text
discount numeric ...
c001 Tiptop Duluth 10
c002 Tyrell Dallas 12
c003 Allied Dallas 8
c004 ACME Duluth 8.5
c005 Weyland Risa 0
c006 ACME Kyoto 0
10:49 PM
1/23/2017


pgAdmin 4
File
Object
Tools
Help
Browser
Servers (1)
PostgreSQL 9.6
Databases (2)
CAP4
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Query- untitled
No limit
CAP4 on postgres@PostgreSQL 9.6
select * from Agents;
Data Output
Explain
Messages
History
aid character
name text
city text
commissi... numeric ...
a01 Smith New York 6.5
a02 Jones Newark 6
a03 Perry Tokyo 7
a04 Grey New York 6
a05 Otasi Duluth 5
a06 Smith Dallas 5
a08 Bond London 7.07
10:53 PM
1/23/2017


pgAdmin 4
File
Object
Tools
Help
Browser
Servers (1)
PostgreSQL 9.6
Databases (2)
CAP4
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Query- untitled
No limit
CAP4 on postgres@PostgreSQL 9.6
150
151
152
153
154
155
156
157
158
159
160 select * from Products;
161
Data Output
Explain
Messages
History
pid character
name text
city text
quantity integer
priceusd numeric ...
p01 comb Dallas 111400 0.5
p02 brush Newark 203000 0.5
p03 razor Duluth 150600 1
p04 pen Duluth 125300 1
p05 pencil Dallas 221400 1
p06 trapper Dallas 123100 2
p07 case Newark 100500 1
p08 eraser Newark 200600 1.25
10:55 PM
1/23/2017


pgAdmin 4
File
Object
Tools
Help
Browser
Servers (1)
PostgreSQL 9.6
Databases (2)
CAP4
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Query- untitled
No limit
CAP4 on postgres@PostgreSQL 9.6
157
158
159
160
161 select * from Orders;
162
Data Output
Explain
Messages
History
ordnumb... integer
month character
cid character
aid character
pid character
qty integer
totalusd numeric ...
1011 Jan c001 a01 p01 1000 450
1012 Jan c002 a03 p03 1000 880
1015 Jan c003 a03 p05 1200 1104
1016 Jan c006 a01 p01 1000 500
1017 Feb c001 a06 p03 600 540
1018 Feb c001 a03 p04 600 540
1019 Feb c001 a02 p02 400 180
1020 Feb c006 a03 p07 600 600
1021 Feb c004 a06 p01 1000 460
1022 Mar c001 a05 p06 400 720
1023 Mar c001 a04 p05 500 450
1024 Mar c006 a06 p01 800 400
1025 Apr c001 a05 p07 800 720
1026 May c002 a05 p03 800 744
10:56 PM
1/23/2017

Key Distinctions

The primary key declares the unique column in a table. It can be declared after the data type of the attribute, or at the end of all the attributes. It cannot have a NULL value. UNIQUE is synonymous and can be used interchangeably with primary key. An example would be cid in the Customers table.

A super key is a column or set of columns that ensures that every row will be unique. An example would be name, city and priceUSD in the Products table.

A candidate key is a superkey with the minimal number of columns. An example would be aid in the Agents table.

A foreign key is a value in a one table that must match the primary key of another table. An example is the foreign key pid in the Orders table that matches the primary key pid in the Products table.

Data Types

All attributes must have a data type. According to our text book, there are six primitive data types which I have listed below. According to the documentation of PostgreSQL, there are many, many more.

1. Character strings
    a. CHAR(n) – character strings that are fixed length up to n characters. Short strings are padded to make n characters.
    b. VARCHAR(n) – character strings that imply a string length is used.
2. Bit strings
    a. BIT(n) – bits of strings of length n.
    b. BIT VARYING(n) – bits of strings of length up to n.
3. BOOLEAN – denotes a logic with the possible values being: TRUE, FALSE, or UNKNOWN.
4. Integer values
    a. INT or INTEGER – typical integer values.
    b. SHORTINT – the number of bits permitted depends on the implementation.
5. Floating-point numbers
    a. FLOAT or REAL – typical floating-point numbers.
    b. DOUBLE PRECISION – used to obtain a higher precision number.
    c. DECIMAL(n,d) or NUMERIC(n,d) – values consist of n decimal digits, with decimal point d positions from the right.
6. Dates and Times - character strings in a special form.
    a. DATE – for example ‘YYYY-MM-DD’.
    b. TIME – for example ‘HH:MM:SS’.

DVDs That I Own Table

title	text	not NULL,
genre	text,	
year	INT	not NULL,
maleStar	text,	
femaleStar	text,	
length	INT	not NULL,
description	text,	
lastWatched	DATE,	

PRIMARY KEY (title, year);

Relational Rules

The "first normal form" rule (1NF) is that all columns must be atomic or indivisible. Each row must be unique, as duplications can cause problems. For every row-by-column position there can be only one value, no structure. If there were a list of values, there would be no easy way to manipulate them and the data would be much harder to retrieve.

This example would make it very difficult o find out how many hammers were sold:

orderId	customerId	items
1	4	5 hammer, 3 wrench
2	23	1 hammer
3	15	2 wrench, 1 key

The "access rows by content only" rule. If you try to point at a row-by-column position by using column1, row A for example, you most likely won't get the data you were looking for. For one thing, you will only get the information that is in column1, row A. In the above example, the result would be 1 if you were querying the snapshot. You must ask for what is there not where it is, because the data may not be in the same position anymore in the database as it is in the snapshot.

The "all rows must be unique" rule. In theory, you can't have two rows with all the same data in it. This ensures that duplication and inconsistency are avoided. This is the rule that is most often violated. If you have for example:

customerID	name	City
1	Bob Smith	Poughkeepsie
2	Bob Smith	Hyde Park

This could be two different Bob Smith's or he could have moved and someone added another record for him. How would you know without a lot more data? How would you know which one was correct or if they both are?