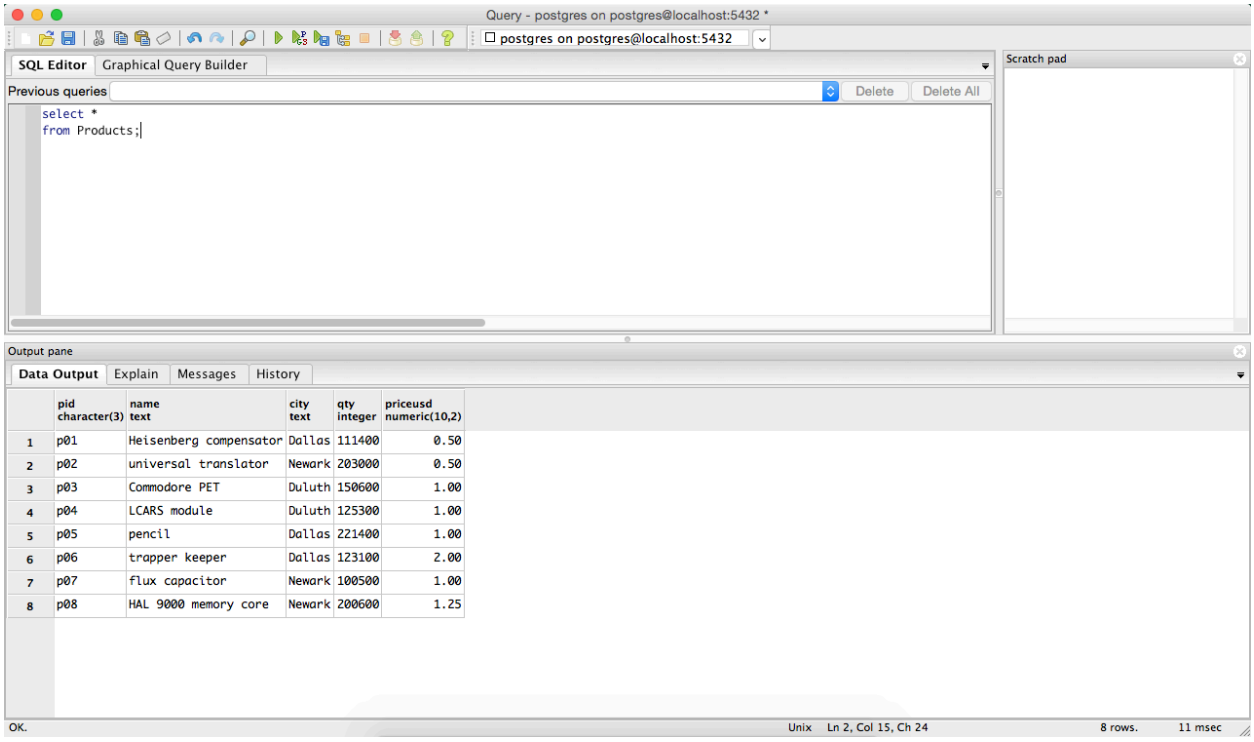
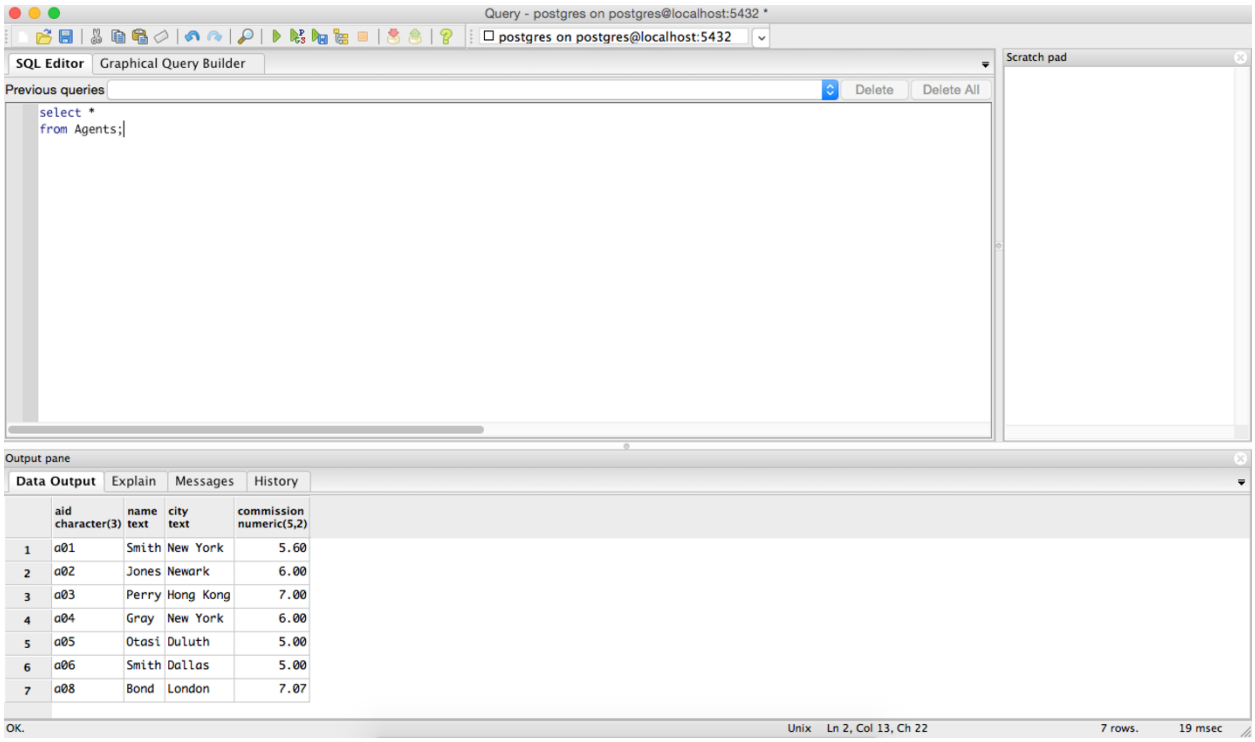
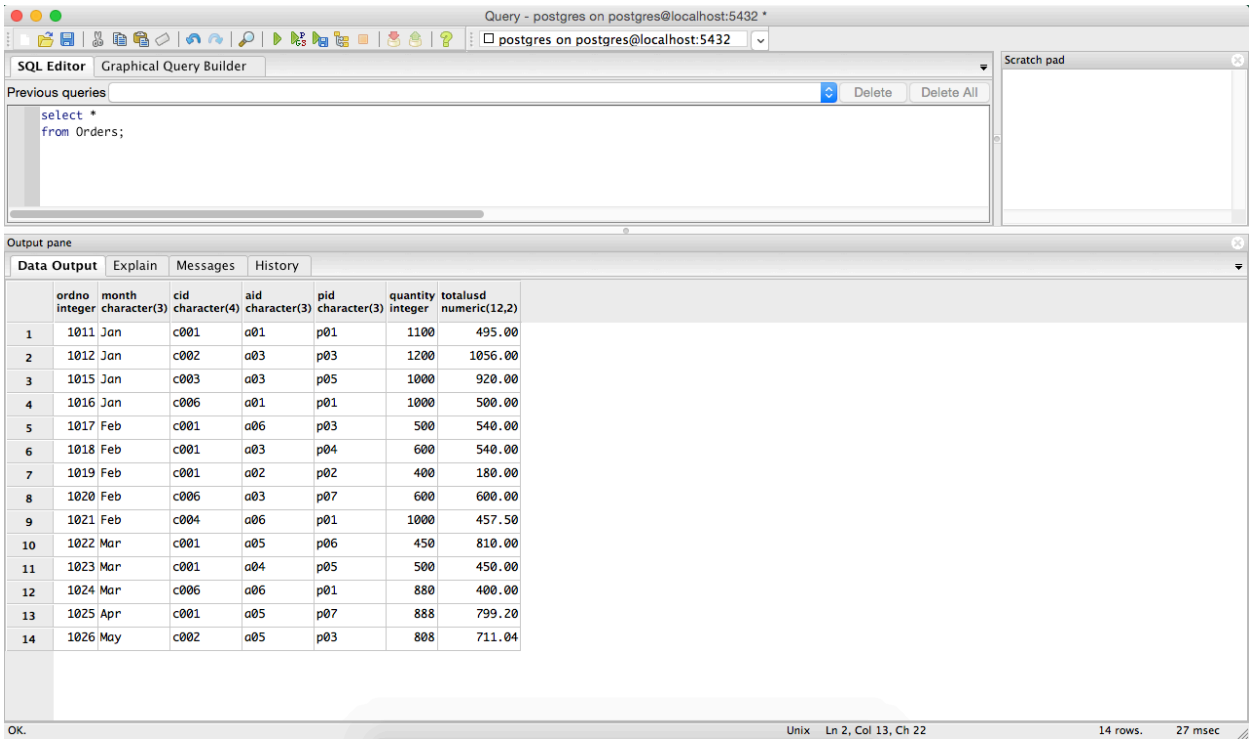
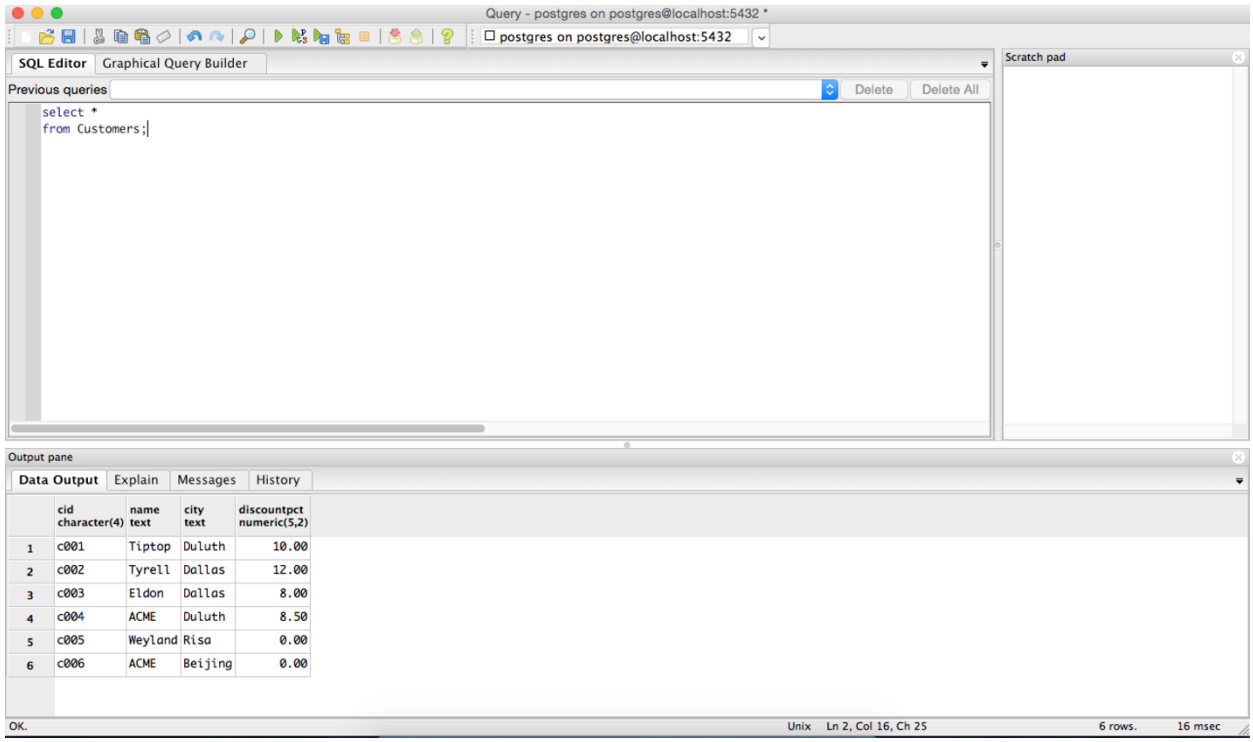
1.)


Query - postgres on postgres@localhost:5432 *
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
select *
from Products;
Output pane
Data Output
Explain
Messages
History
pid character(3)
name text
city text
qty integer
priceusd numeric(10,2)
1 p01 Heisenberg compensator Dallas 111400 0.50
2 p02 universal translator Newark 203000 0.50
3 p03 Commodore PET Duluth 150600 1.00
4 p04 LCARS module Duluth 125300 1.00
5 p05 pencil Dallas 221400 1.00
6 p06 trapper keeper Dallas 123100 2.00
7 p07 flux capacitor Newark 100500 1.00
8 p08 HAL 9000 memory core Newark 200600 1.25
OK.
Unix Ln 2, Col 15, Ch 24
8 rows.
11 msec


Query - postgres on postgres@localhost:5432 *
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
select *
from Agents;
Output pane
Data Output
Explain
Messages
History
aid character(3)
name text
city text
commission numeric(5,2)
1 a01 Smith New York 5.60
2 a02 Jones Newark 6.00
3 a03 Perry Hong Kong 7.00
4 a04 Gray New York 6.00
5 a05 Otasi Duluth 5.00
6 a06 Smith Dallas 5.00
7 a08 Bond London 7.07
OK.
Unix Ln 2, Col 13, Ch 22
7 rows.
19 msec


Query - postgres on postgres@localhost:5432 *
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
select *
from Orders;
Output pane
Data Output
Explain
Messages
History
ordno integer
month character(3)
cid character(4)
aid character(3)
pid character(3)
quantity integer
totalusd numeric(12,2)
1 1011 Jan c001 a01 p01 1100 495.00
2 1012 Jan c002 a03 p03 1200 1056.00
3 1015 Jan c003 a03 p05 1000 920.00
4 1016 Jan c006 a01 p01 1000 500.00
5 1017 Feb c001 a06 p03 500 540.00
6 1018 Feb c001 a03 p04 600 540.00
7 1019 Feb c001 a02 p02 400 180.00
8 1020 Feb c006 a03 p07 600 600.00
9 1021 Feb c004 a06 p01 1000 457.50
10 1022 Mar c001 a05 p06 450 810.00
11 1023 Mar c001 a04 p05 500 450.00
12 1024 Mar c006 a06 p01 880 400.00
13 1025 Apr c001 a05 p07 888 799.20
14 1026 May c002 a05 p03 808 711.04
OK.
Unix Ln 2, Col 13, Ch 22
14 rows.
27 msec


Query - postgres on postgres@localhost:5432 *
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
select *
from Customers;
Output pane
Data Output
Explain
Messages
History
cid character(4)
name text
city text
discountpct numeric(5,2)
1 c001 Tiptop Duluth 10.00
2 c002 Tyrell Dallas 12.00
3 c003 Eldon Dallas 8.00
4 c004 ACME Duluth 8.50
5 c005 Weyland Risa 0.00
6 c006 ACME Beijing 0.00
OK.
Unix Ln 2, Col 16, Ch 25
6 rows.
16 msec

2.)

A super key is a field (attribute/column) or set of fields that uniquely identify every row in a table. For example, the super key of a table containing superhero information with the attributes ID, first name, last name, and superhero name would be those attribute names. Candidate key is the minimal super key that contains the least number of columns possible to distinguish between the rows. For example, if all that is needed to differentiate between the rows is ID, then it is encouraged to throw away as much as possible while also preserving uniqueness. For example, if the table was about class scheduling at Marist College, the candidate key might have to be both the class time and classroom location attributes because those together would help to tell the different classes apart. However, if the table was about different customers, their ID number could be the candidate key because each would have a different one. Finally, primary keys are a form of a candidate key that the database designer chooses to uniquely identify each row. ID number is commonly used in this case to distinctly separate the different rows.

3.)

There are numerous data types that are supported by SQL. There are CHAR and VARCHAR, BIT and BIT VARYING, BOOLEAN, INT or INTEGER, FLOAT or REAL, NUMERIC, and DATE and TIME. CHAR and VARCHAR denote fixed-length strings of up to n characters. BIT and BIT VARYING are like CHAR and VARCHAR in that they represent strings of fixed and varying lengths. However, instead of characters, these data types denote bit strings. BOOLEAN signifies a value that is logical. For example, the value could be TRUE, FALSE, or UNKOWN. INT involves typical integer values. FLOAT or REAL are associated

with classic floating point numbers (eg. 0123.45). NUMERIC values are any that can be measured. Finally, DATE and TIME are just strings of a special form. These values can be represented as string types, or may be designated as data and time, depending on what makes sense.

An example of a table that may be created could be a student table, containing data regarding different students in a high school math class. The name of this table is Mrs. Dunn's 11^{th} Grade Math. Some of the columns could be student ID number, student name, HW1, HW2, Project1, Lab1, Midterm, and Final. The student ID number would be of the INT data type, and cannot be null. It is the primary key in this table and must have a value for all new rows. Student name would be a CHAR data type because it contains all characters, and cannot be null. Every student in each row must have a student ID and student name. The rest of the values are NUMERIC, and some may be null. This is because a student may not take a test, so their row would be null for that test.

4.)

The "first normal form" rule basically states that all values in a database table (with rows and columns) must contain atomic values only. This means that the values are indivisible and cannot be broken down into smaller portions. For example, if the column held values of locations of business, the row cannot contain a value of Chicago: New York City: California. It must only contain one value. One column could be Location1 and could contain Chicago. Another could be Location2 and contain New York City, and so on. An alternative option to this could be create a separate table for locations, and then matching the location codes from the separate table and

putting them in a third table titled “businessLocations”. For database normalization to occur, the first normal form is imperative and is a minimum requirement to improve the database structure.

The access rows by content only rule implies that you cannot refer to a row in a location such as “third one down” because the rows do not actually have order. The tables in a database are set, and sets are unordered. The cells have no intrinsic ordinal values. Cells are called upon by the value that they contain. This is incredibly important when querying because there is no way in the language to ask to retrieve the third row down because that does not exist. It would have to ask for a value that is in a row to call from that spot.

Finally, the “all rows must be unique” rule states that two rows cannot be identical in every attribute. This would make them the same row, and leads to duplicate values. This is important because in the querying language we want the ability to address every individual row if needed, and with two rows containing the same exact information, this ability is mitigated. A way to resolve this is by ensuring that there are primary keys in place that create this uniqueness between the rows, and help to differentiate between information, or that another table is added if need be.