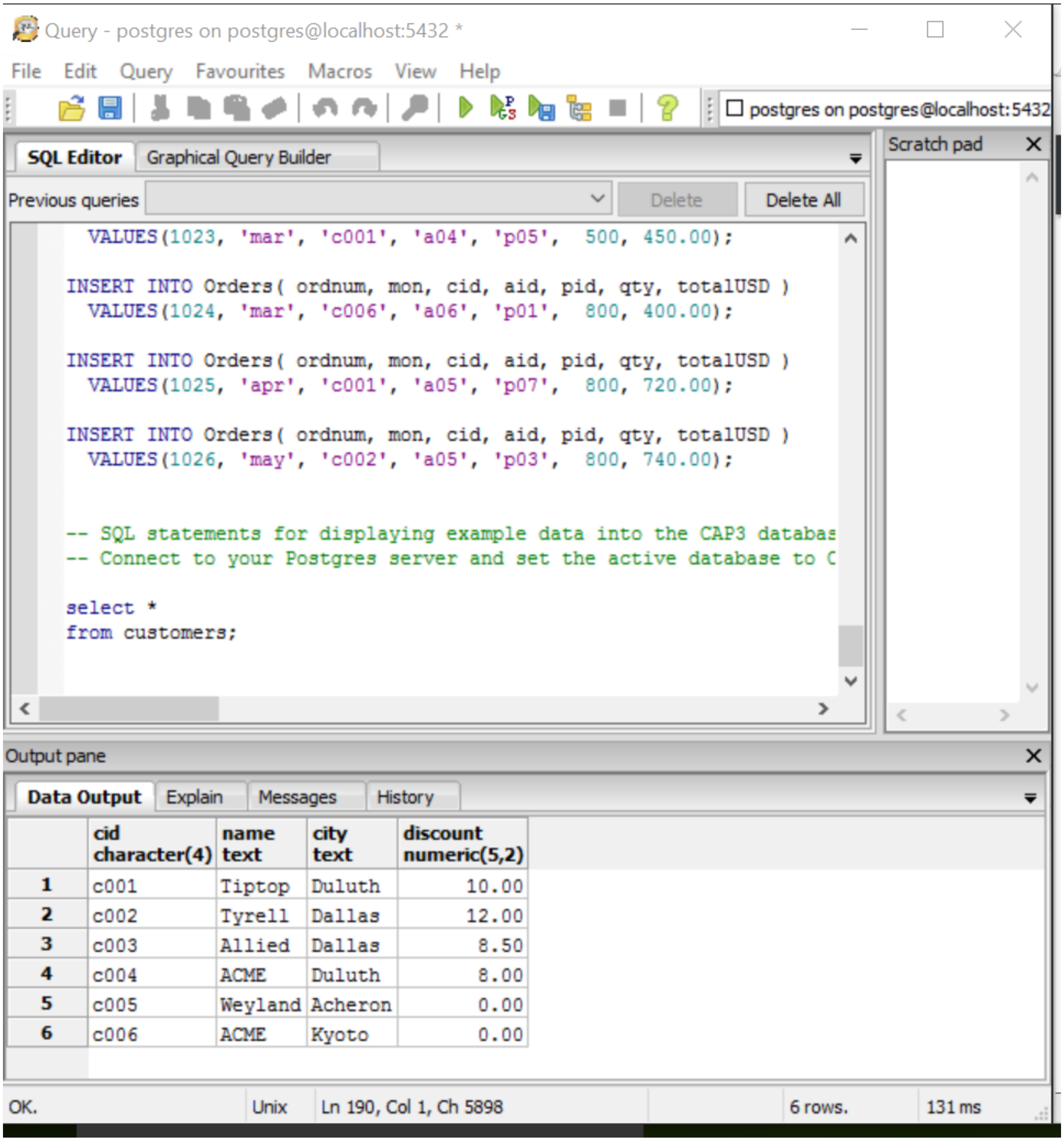
Liliya Yerassilova   Lab 2   01/30/2016



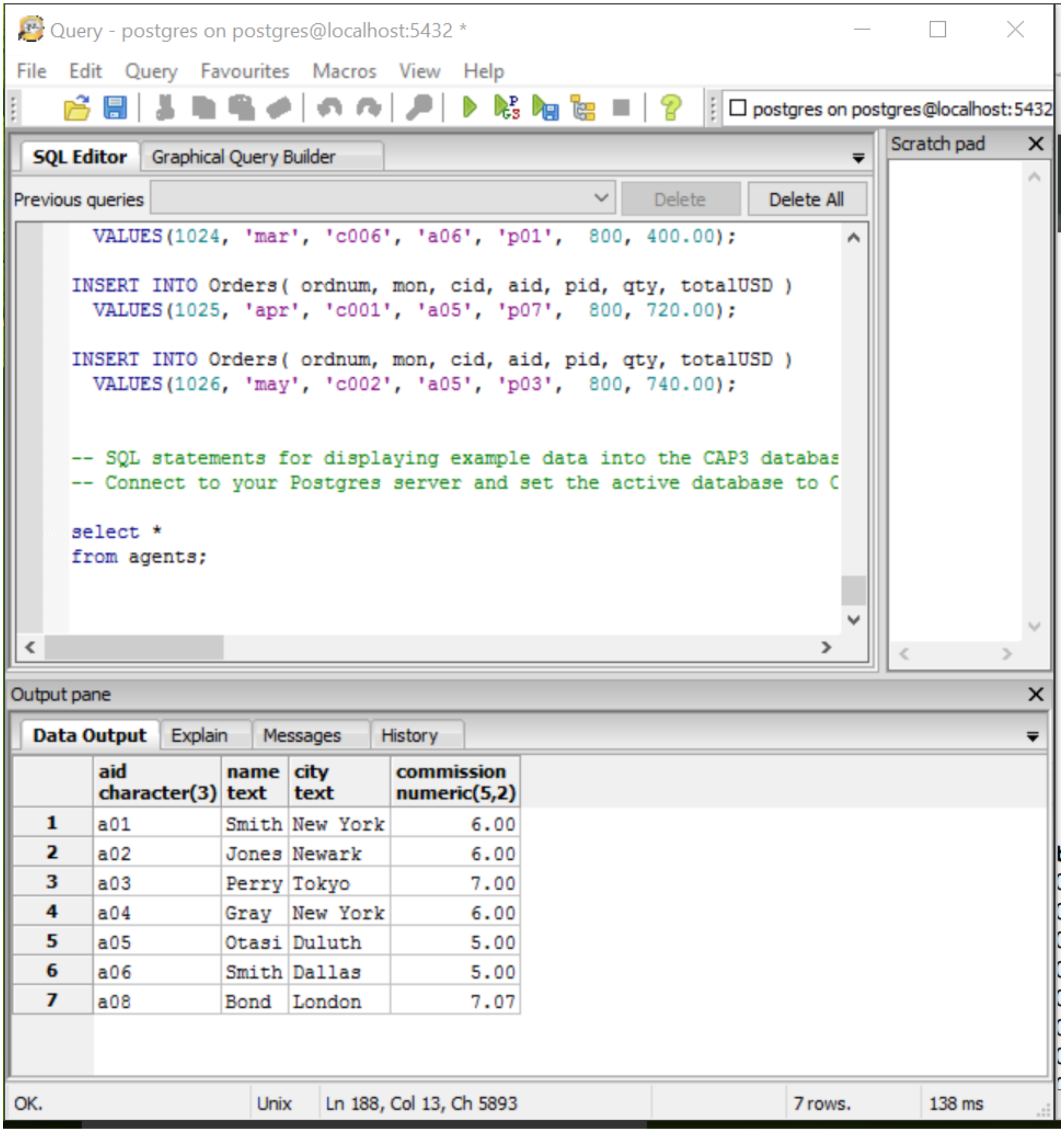
Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
  VALUES(1023, 'mar', 'c001', 'a04', 'p05',  500, 450.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1024, 'mar', 'c006', 'a06', 'p01',  800, 400.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1025, 'apr', 'c001', 'a05', 'p07',  800, 720.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1026, 'may', 'c002', 'a05', 'p03',  800, 740.00);


-- SQL statements for displaying example data into the CAP3 databas
-- Connect to your Postgres server and set the active database to C

select *
from customers;
```

	cid character(4)	name text	city text	discount numeric(5,2)
1	c001	Tiptop	Duluth	10.00
2	c002	Tyrell	Dallas	12.00
3	c003	Allied	Dallas	8.50
4	c004	ACME	Duluth	8.00
5	c005	Weyland	Acheron	0.00
6	c006	ACME	Kyoto	0.00

OK.   Unix   Ln 190, Col 1, Ch 5898   6 rows.   131 ms

Liliya Yerassilova   Lab 2   01/30/2016



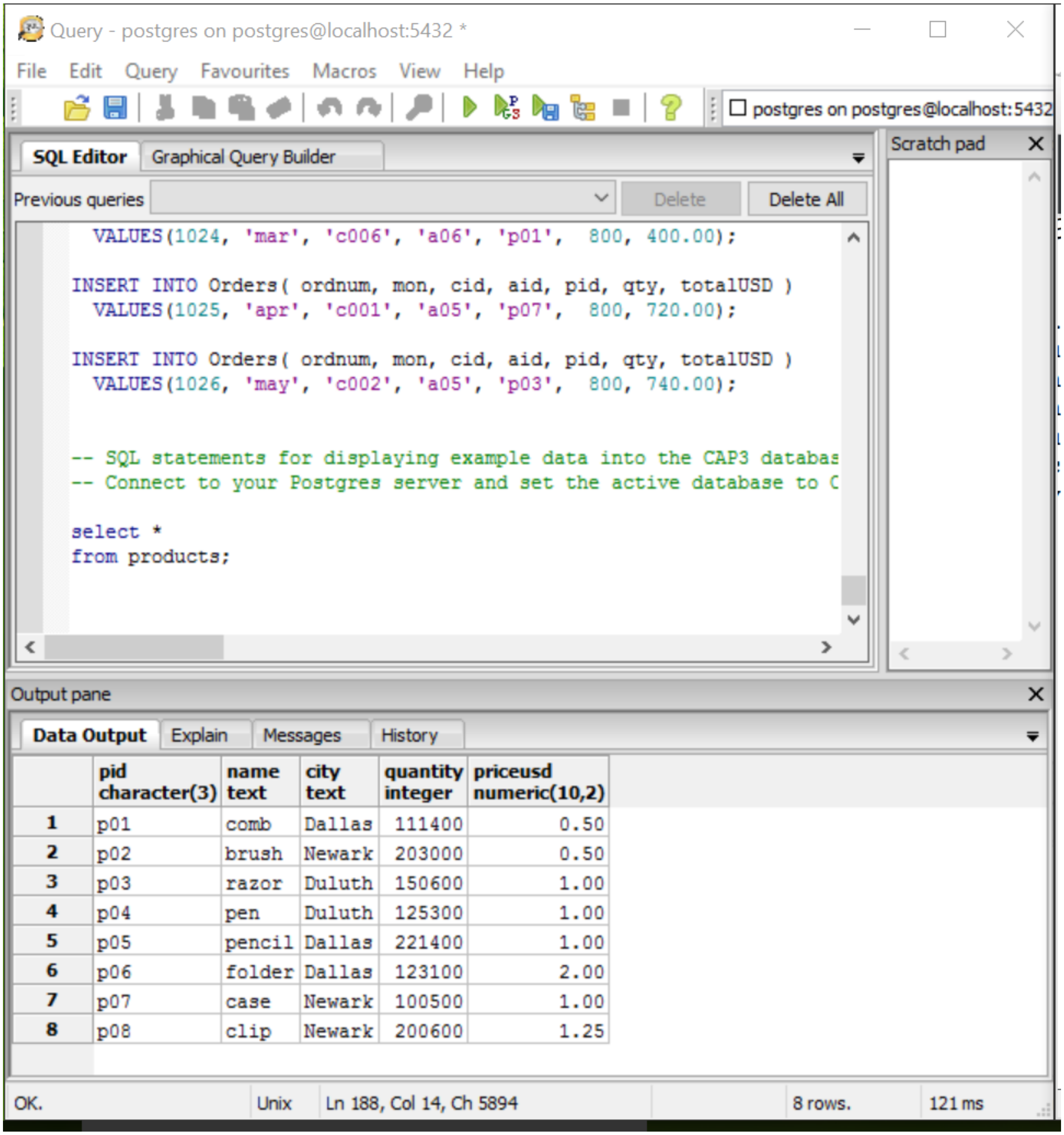
Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
□ postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Scratch pad
Previous queries
Delete
Delete All
Output pane
Data Output
Explain
Messages
History

```
  VALUES(1024, 'mar', 'c006', 'a06', 'p01',  800, 400.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1025, 'apr', 'c001', 'a05', 'p07',  800, 720.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1026, 'may', 'c002', 'a05', 'p03',  800, 740.00);


-- SQL statements for displaying example data into the CAP3 databas
-- Connect to your Postgres server and set the active database to C

select *
from agents;
```

	aid character(3)	name text	city text	commission numeric(5,2)
1	a01	Smith	New York	6.00
2	a02	Jones	Newark	6.00
3	a03	Perry	Tokyo	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 188, Col 13, Ch 5893 7 rows. 138 ms

Liliya Yerassilova Lab 2 01/30/2016

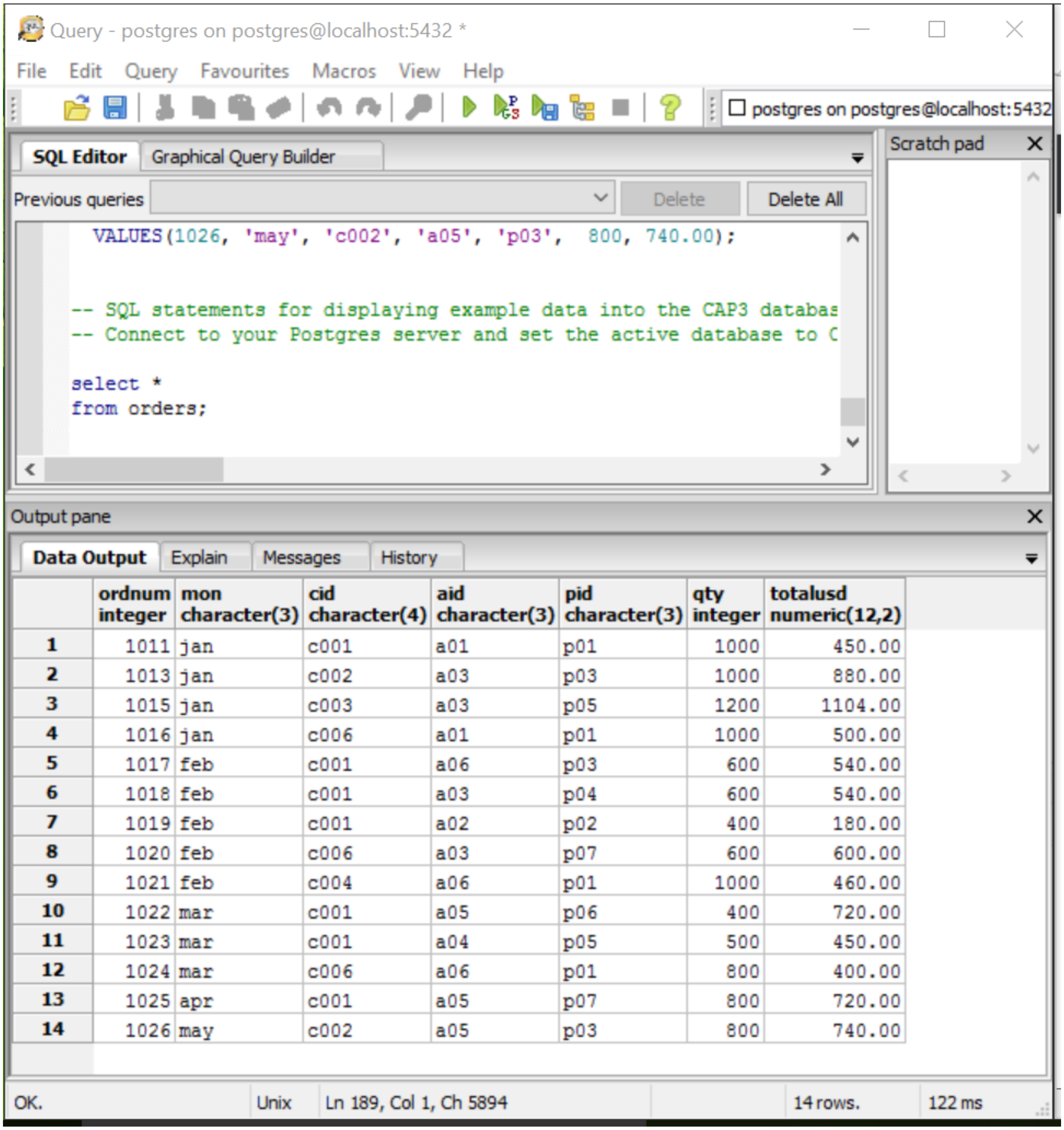

Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
  VALUES(1024, 'mar', 'c006', 'a06', 'p01',  800, 400.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1025, 'apr', 'c001', 'a05', 'p07',  800, 720.00);

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
  VALUES(1026, 'may', 'c002', 'a05', 'p03',  800, 740.00);


-- SQL statements for displaying example data into the CAP3 databas
-- Connect to your Postgres server and set the active database to C

select *
from products;
```

	pid character(3)	name text	city text	quantity integer	priceusd numeric(10,2)
1	p01	comb	Dallas	111400	0.50
2	p02	brush	Newark	203000	0.50
3	p03	razor	Duluth	150600	1.00
4	p04	pen	Duluth	125300	1.00
5	p05	pencil	Dallas	221400	1.00
6	p06	folder	Dallas	123100	2.00
7	p07	case	Newark	100500	1.00
8	p08	clip	Newark	200600	1.25

OK. Unix Ln 188, Col 14, Ch 5894 8 rows. 121 ms

Liliya Yerassilova Lab 2 01/30/2016


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
☐ postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
    VALUES(1026, 'may', 'c002', 'a05', 'p03',  800, 740.00);


-- SQL statements for displaying example data into the CAP3 databas
-- Connect to your Postgres server and set the active database to C

select *
from orders;
```

	ordnum integer	mon character(3)	cid character(4)	aid character(3)	pid character(3)	qty integer	totalusd numeric(12,2)
1	1011	jan	c001	a01	p01	1000	450.00
2	1013	jan	c002	a03	p03	1000	880.00
3	1015	jan	c003	a03	p05	1200	1104.00
4	1016	jan	c006	a01	p01	1000	500.00
5	1017	feb	c001	a06	p03	600	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	460.00
10	1022	mar	c001	a05	p06	400	720.00
11	1023	mar	c001	a04	p05	500	450.00
12	1024	mar	c006	a06	p01	800	400.00
13	1025	apr	c001	a05	p07	800	720.00
14	1026	may	c002	a05	p03	800	740.00

OK. Unix Ln 189, Col 1, Ch 5894 14 rows. 122 ms

2) Explain the distinctions among the terms primary key, candidate key, and superkey

Primary key is a super key that you choose to be primary. It is the main reference key in a table. In a student database, a student id would serve well as a primary key because it is minimal (only one value) and can uniquely identify each student.

Candidate key – minimal super key with the fewest possible number of columns that still uniquely identify them. It is a subset of a super key. For example, in a registrar database, a student Social Security number can be a candidate key as well as a student id number. However, student id would, probably, serve as a primary key since international students might not have a Social security number, in which case the key will no longer uniquely identify the student in the database.

Super key is any field (column) or a set of fields (columns) that uniquely identify any row in a table. For example, students' first and last names together can uniquely identify students in a database, however, separately, neither first name nor last name uniquely identify students. Only the combination can be used for identification purposes. Super key does not have to satisfy the minimality condition that candidate and primary keys have to.

3)

There are two types of data: generic and numerator. Generic data types include different type of data, such as strings, text, integers, characters, etc. Numeric data types only take the values that you specify yourself, such as gender, day of a week, month. We can create a database for the chemical elements of the periodic table. Even though the periodic table already very well classifies the elements, a database can hold more information about each element and provide an easier access to all the extra information. Table called "Chemical Elements" would have columns for the Name, Symbol, Atomic, number, Atomic weight, boiling point, number of isotopes, and year discovered.

Name column would have text data type, which is generic data, this is not nullable. Symbol would be also generic, a string, not nullable. Atomic number would be generic, but an integer, whereas, atomic weight should include floating numbers, neither of these is nullable. Boiling point should specify the scale in the title and should include floating numbers as well. This element can be nullable sometimes when boiling point in unknown. For example, Bohrium, a synthetic element that is not found in nature, is a radioactive element whose boiling point is unknown. Number of isotopes should be an integer and can be nullable for the elements that don't have any isotopes. Year discovered is generic data type, an integer.

For an example of a numerator data type, we can include some other columns, such as "Found in nature", data type for this would be a Boolean (True or False only). Another example for this topic would be to include "Radioactive" column that would contain Yes/No data type, which is a numerator type of data.

4)

a. First Normal Rule says that columns with multivalued intervals are not allowed. This means that for each column there has to be only one piece of data, columns cannot contain an entire record. The data has to be atomic, the lowest possible unit. For example, in the chemical elements database, the table cannot have a column "Discoverers" and contain all the people who discovered an element. There has to be only one name, if such column exists. If there were more than one person for the column, then it would be impossible to query the data.

b. Accessing rows by content only means that when querying, we have to specify which row we are interested in by giving exact data or description of it, not its order number. Since tables are sets, there are no orders, therefore, we cannot expect to receive data from a query that specifies a number order of the row. From the elements table, we cannot query by specifying that we need information on the first or second row. We have to specify which element we are interested in by clarifying its content, such as its name, symbol, or atomic weight.

c. The last rule declares that all rows must be unique, duplicate rows are not allowed. We cannot have the exact same row for the same elements. For example, if we had to include isotopes themselves as separate elements in the above database, we would have most of the other data the same (since isotopes are basically the same element, except they differ in atomic weight). Since this is not allowed, we will need to add a column for “Number of neutrons”, which will differ for all the elements including their isotopes. Duplicate rows will not yield correct results for queries and it is just abundant and irrational to duplicate data.