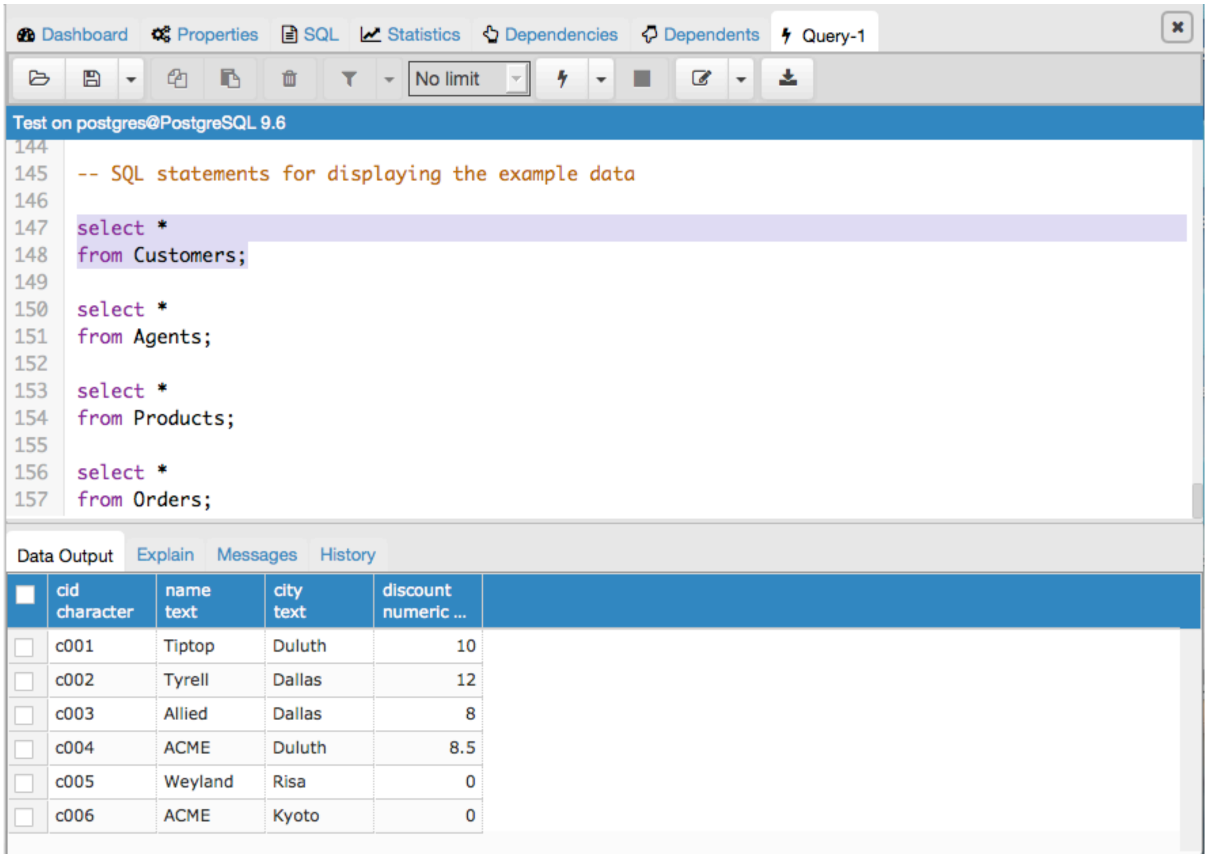
Federica Bruno
308 Database Management
Alan Labouseur
January 31, 2017

Lab 2: CAP Database

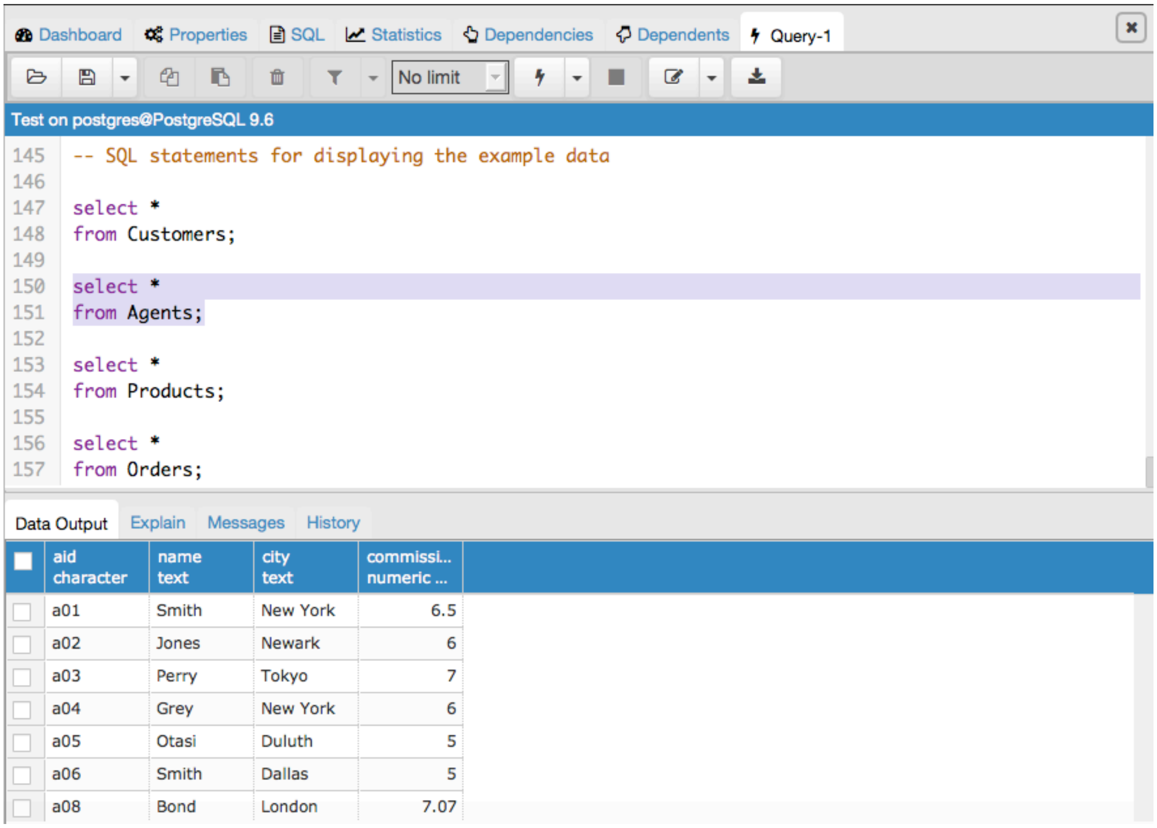
1. **select * from Customers;**



```

-- SQL statements for displaying the example data

select *
from Customers;

select *
from Agents;

select *
from Products;

select *
from Orders;
```

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

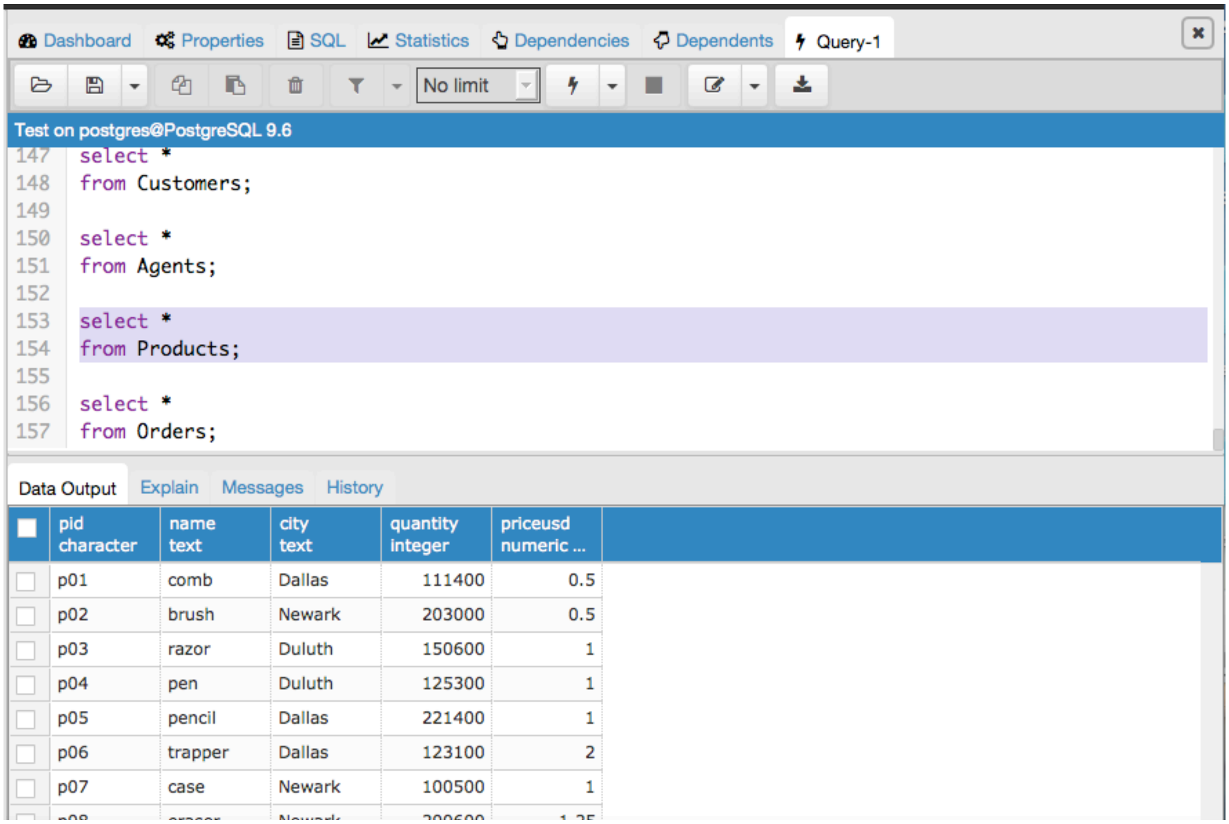
2. select * from Agents;



```
-- SQL statements for displaying the example data

select *
from Customers;

select *
from Agents;

select *
from Products;

select *
from Orders;
```

☐	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

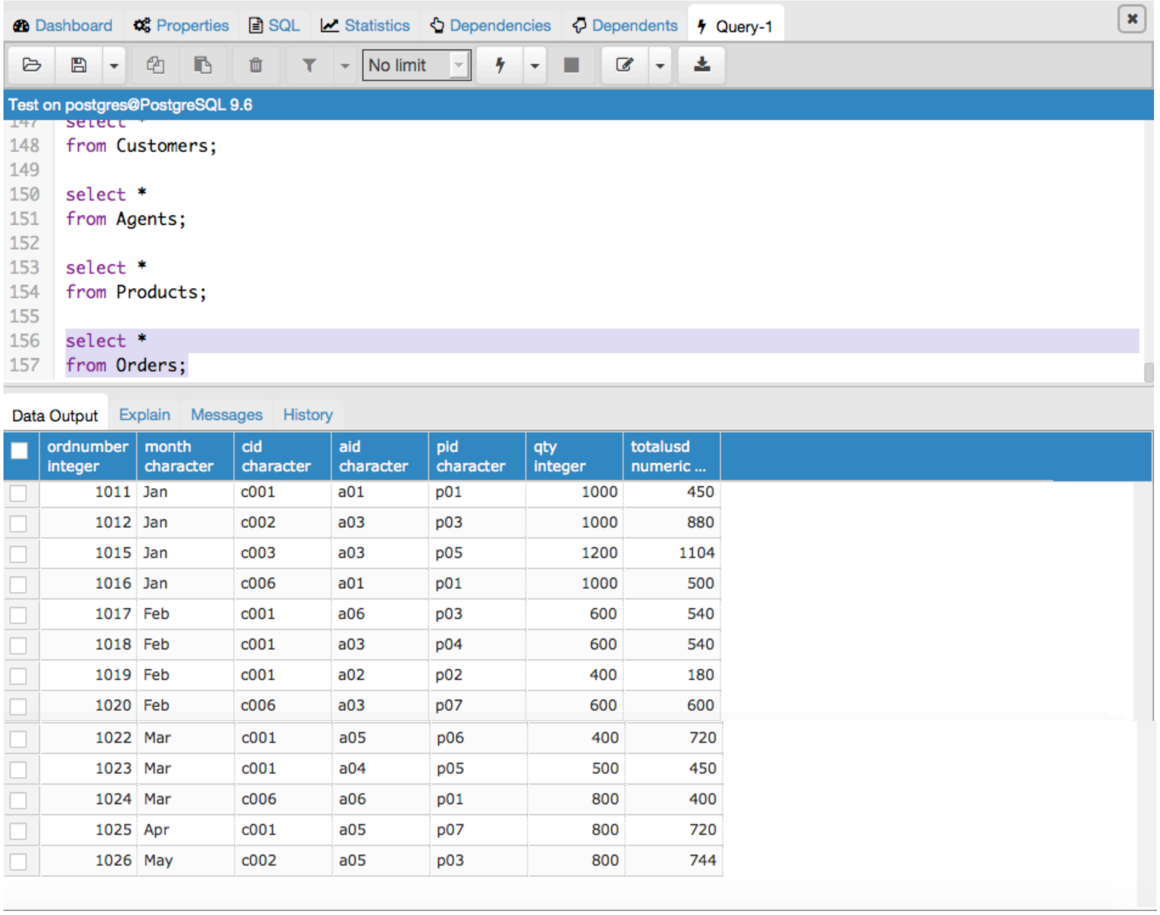
3. select * from Products;



```
select *
from Customers;

select *
from Agents;

select *
from Products;

select *
from Orders;
```

☐	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

4. select * from Orders;



Test on postgres@PostgreSQL 9.6

```
select *
from Customers;

select *
from Agents;

select *
from Products;

select *
from Orders;
```

ordnumber 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
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

2. The super key is a combination of columns that ensure every row is unique. The candidate key is the minimal super key that uniquely identifies every single row in the fewest number of columns, or least amount in length. The primary key is the chosen candidate key to uniquely identify every row of the given table.

3. Data types define the kind of value a column can contain. Each column in a table is required to have a label and data type for that label. The following are some of the general types of data types that are stored within columns of tables: `CHARACTER, VARCHAR, BINARY, BOOLEAN, VARBINARY, INTEGER, SMALLINT, BIGINT, DECIMAL, NUMERIC, FLOAT, REAL, DATE, TIME, TIMESTAMP`, and the list goes on.

   Suppose you create a table named ‘Books’, and within the table you create 5 different fields or columns. The columns are as follows: `(1) ‘Author’ varchar(128); null (2) ‘Title’ varchar(128); null (3) ‘Category’ varchar(16); null (4)‘Year’ smallint; null (5) ‘ISBN’  char(13) not null`.

   Author, title, and category have `VARCHAR` as their data type. `VARCHAR` stands for “variable length character string” and it is suitable for these fields because their values differ in length. The numeric value in parenthesis represent the maximum length a string in that field can contain. ISBN numbers, on the other hand, are a set string containing 13 numbers, and why `CHAR` is suitable for this field, because it has a predicable value. All the fields except for ISBN are null because those fields are unknown for now until information is added to the table, and it ensures that every column has a value whether it is applicable at the moment or not. ISBN is not null because it represents as the primary key of that table.

4. The first normal form rule has two main principles. (1) Information in a database should be stored in tables, in which are made up of rows and columns, and that there should be a primary key that uniquely indentifies each row. (2) A column cannot contain lists of values, but instead must contain atomic values; that is, a value cannot be subdivided into further values but must be a single value. For example, if there is a column that represents ISBNs within a table, you cannot list multiple ISBNs separated by commons in a single row, but each ISBN must be located in a different row. There cannot be repeating groups of columns as well. For instance, if you want to list customer names, you cannot have columns such as CustomerName1, CustomerName2, CustomerName3, etc., but rather must combine each customer name into one column called Customers, and represent each name as a different row. 1NF may be tedious task, but it makes query searches efficient and eliminates duplication. If databases allowed lists within columns, it will make queries very confusing and difficult and slow.

The second relational rule is “Access rows by content rule”. For example, one should not query, “Third one down from the top”, but rather query a specific value or item. Essentially, the backbone principle of this rule is that rows should not contain any order. The reason is because when new information is added, rows may shift and change. Therefore, you should always specify *want* you want and not *where* it is positioned; “what not where.”

The third relational rule is “All rows must be unique”, which seems to be a given. This rule ensures that there will be no duplicating values. This is important because if there happens to be duplicating data that contain errors, that means that there are errors present in more than one location. One may correct the error in one row or location, but may bypass the other. This leads to inconsistent and incompatible data, which may be dangerous when dealing with sensitive material.