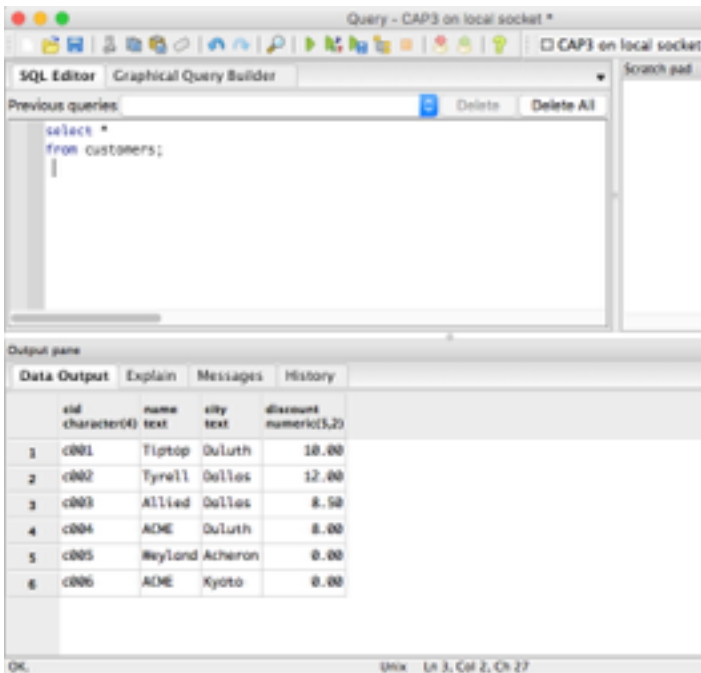


1. Execute the following queries (one at a time) from pgAdmin's SQL Tool:

```
select *
from customers;
select *
from agents;
select *
from products;
select *
from orders;
```

Take a screen shot of each query and its results.



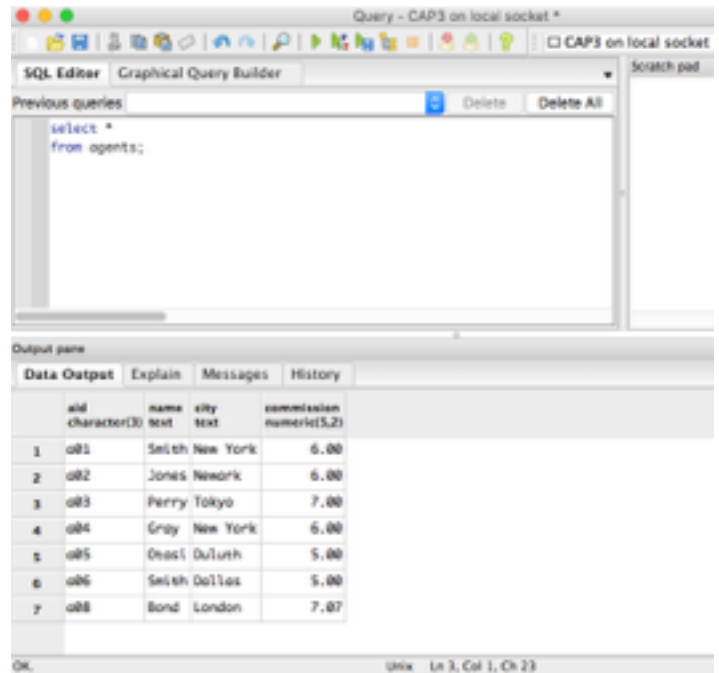
Query - CAP3 on local socket \*

SQL Editor: `select *  
from customers;`

Output pane:

	aid character(4)	name text	city text	discount numeric(5,2)
1	c001	Tiptop	Duluth	10.00
2	c002	Tynell	Dallas	12.00
3	c003	Allied	Dallas	8.50
4	c004	ADME	Duluth	8.00
5	c005	Wayland	Acherron	0.00
6	c006	ADME	Kyoto	0.00

OK. Unix Ln 3, Col 2, Ch 27



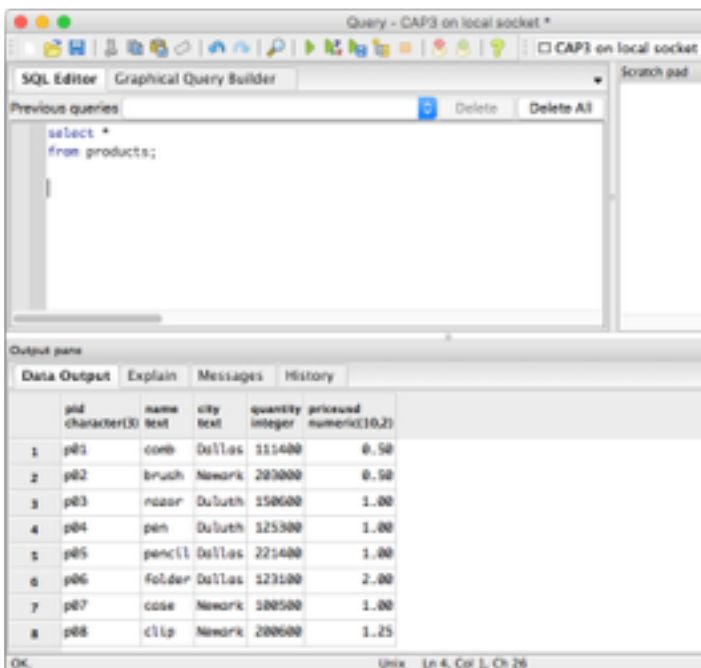
Query - CAP3 on local socket \*

SQL Editor: `select *  
from agents;`

Output pane:

	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	Osasi	Duluth	5.00
6	a06	Smith	Dallas	5.00
7	a08	Bond	London	7.07

OK. Unix Ln 3, Col 1, Ch 23



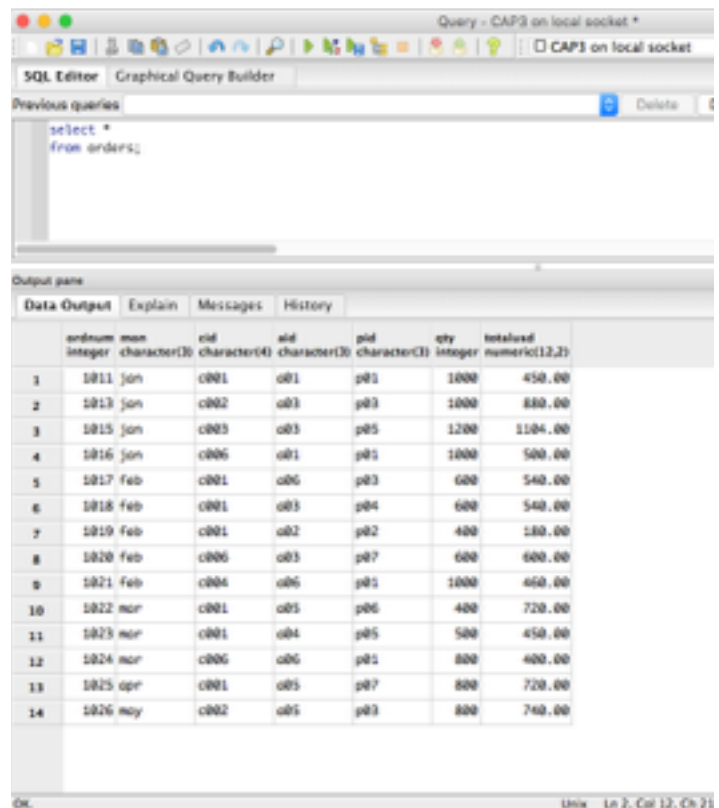
Query - CAP3 on local socket \*

SQL Editor: `select *  
from products;`

Output pane:

	pid character(3)	name text	city text	quantity integer	pricesold numeric(10,2)
1	p01	comb	Dallas	111400	0.50
2	p02	brush	Newark	203000	0.50
3	p03	razor	Duluth	150000	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	200000	1.25

OK. Unix Ln 4, Col 1, Ch 26



Query - CAP3 on local socket \*

SQL Editor: `select *  
from orders;`

Output pane:

	ordnum integer	man character(3)	aid character(4)	pid character(3)	qty integer	totalused numeric(12,2)
1	1011	jon	c001	p01	1000	450.00
2	1013	jon	c002	a01	1000	880.00
3	1015	jon	c003	a03	1200	1104.00
4	1016	jon	c006	a01	1000	500.00
5	1017	feb	c001	a06	600	540.00
6	1018	feb	c001	a03	600	540.00
7	1019	feb	c001	a02	400	180.00
8	1020	feb	c006	a03	600	600.00
9	1021	feb	c004	a06	1000	600.00
10	1022	mar	c001	a05	400	720.00
11	1023	mar	c001	a04	500	450.00
12	1024	mar	c006	a06	800	400.00
13	1025	apr	c001	a05	800	720.00
14	1026	may	c002	a05	800	760.00

OK. Unix Ln 2, Col 12, Ch 21

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

A superkey is any column or set of columns that is uniquely defined in every row. A candidate key is a type of superkey which is minimal in the sense that it uses the least amount of columns to uniquely ID every row. A primary key is a type of candidate key that is used to firstly identify each row uniquely.

3. Write a short essay on data types. Select a topic for which you might create a table. Name the table and list its Wields (columns). For each Wield, give its data type and whether or not it is nullable.

Some musical purists (like myself) might create a entire database to keep track of their physical a music collection (CDs, Cassette Tapes, Vinyl Records). So at least one table to every medium would mean at least one table for CDs. The table name would be “PhysicalCDs” and the Wields or columns would be: CDID, Album\_Title, Artist\_Name, Genre, Year, Recording\_Company, Producer, CID or ConditionID, Deluxe, Case\_Included, Lyrics\_Included and SourceID (SID). In context, ConditionID or CID is whether the CD is new, used or unrecognizable kind of like on amazon there will be a conditions table with conditions and descriptions. Deluxe is just indicative of if it is the Deluxe Edition of the Album. Source ID is just supposed to reference another table of people or places CDs are received or bought from.

CDID would be the Primary Key type char of length 4 ('char(4)) not nullable. Album\_Title would be type text and not nullable. Artist\_Name would be type text and not nullable. Genre would be type text and nullable. Year would be type integer and not nullable. Recording\_Company would be type text and not nullable. Producer would be type text and not nullable. CID would be a foreign key of type text and not nullable. Deluxe would be type boolean and not nullable. Case\_Included would be type boolean and not nullable. Lyrics\_Included would be type boolean and nullable. SID would be a foreign key of type text and not nullable.

4. Explain the following relational “rules” with examples and reasons why they are important.

- a. The “First normal form” rule
- b. The “access rows by content only” rule
- c. The “all rows must be unique” rule

a. The “First normal form” rule

The first normal form rule states that the intersection of any row or column should have data. The data must be an unstructured or atomic piece of data. That is there can never be more than more the two distinctly meaningful pieces of data at a distinct intersection of a row and column. So for example if there is a coma separated list of color possibilities for a product in [amazon.com](https://www.amazon.com)’s vast database of products then the rule is violated because despite the simplicity of a comma separated list is structured. The complexity of trying to find all of the rows with only one or only two of those colors is exponentially increased. More over if a certain attribute is supposed to contained structured data then the solution in a relational database is to make a new table and *relate them*. Other solutions often either create an unreasonable amount of extra columns for the same type of data in context or leave a lot of unnecessary empty space which is not only inefficient but wasteful.

b. The “access rows by content only” rule

Allow the question “what?” never the question “where?” So, tables are sets and there for as we know from discrete math, therefore there is order doesn't matter. So when accessing data from a table you can get the information by *what* is in a certain column but not by *where* that column might be. For example, Album might be the second column in a database of iTunes songs but users cannot say give me the songs by the second column. Users instead must access songs and order them by the attribute (column) named “Album.” This is rule is necessary because as relational databases grow and change the location of individual rows and columns will change. However the attributes rows and the information itself will be searchable by their content.

c. The “all rows must be unique” rule

You cannot have duplicate rows. So, the entire row should not be identical to any other. For example, even if two gmail users have the same username (which gmail thankfully does not let them do) if they also have different passwords or account IDs then they are not identical users. However if there are two users with the same username and the same password and the rest of there columns match exactly then there is a serious problem not only will it be hard to

figure out which email to use but also where to send said email when the emails need to be accessed. Also just in terms of storage duplicate rows create redundancy. Databases and redundancy do not mix well that is kind of a big point of the existence of relational databases.