1.

Previous queries

```
  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 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

Output pane

Data Output | Explain | Messages | History

	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

Previous queries

```
  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 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

Output pane

Data Output | Explain | Messages | History

	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

-

Previous queries

```
  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 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

Output pane

Data Output | Explain | Messages | History

	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

Previous queries

```sql
-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

Output pane

Data Output | Explain | Messages | History

	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

2. A superkey is any column or any set of columns that uniquely identify every row in a table. For example, the columns “CWID”, “First Name”, and “Last Name” would be a superkey in a table sorting Marist College students. A candidate key is a superkey with the fewest possible columns inside it. So if we made our superkey to only include “CWID”, we would have a candidate key. A primary key is a superkey we choose to make primary. We would choose the CWID to be the primary identifier that distinguishes every Marist student from each other, rather than First Name or Last Name.

3. Data types are definitions we give to pieces of data based off of what kind of information can be inputted for each. For example, let’s say we wanted to create a table for keeping track of how many red dragons and blue dragons we are selling. We’ll name our table “Dragon Sales” and in that table will be columns for order number, customer name, color dragon ordered, and quantity ordered. The data type for order number would be VarChar, a generic type. The customer name and color dragon ordered would both be text types, and the quantity ordered would be an integer type. None of these types can be nullable.

4a. The “first normal form” rule states that columns with multi-valued attributes or columns with internal structure are not allowed. This means that only one unstructured value can be at the intersection of each column and row. For example, at the intersection of the CWID column and the row for a student, Alvin, there cannot be both ID numbers 200-56-770 and 200-68-911.

4b. The “access rows by content only” rule states that you can only access rows by what they have and not by where they are. For example, to find out a Alvin’s information, you cannot search for the second-to-last row of the table, you have to search for the row where the first name is “Alvin”.

4c. The “all rows must be unique” rule pretty simply states that all rows must be unique, as you may have guessed. If we already have Alvin’s information recorded once on file, we do not need a second account of it, as that would take up space another student’s information could take up.