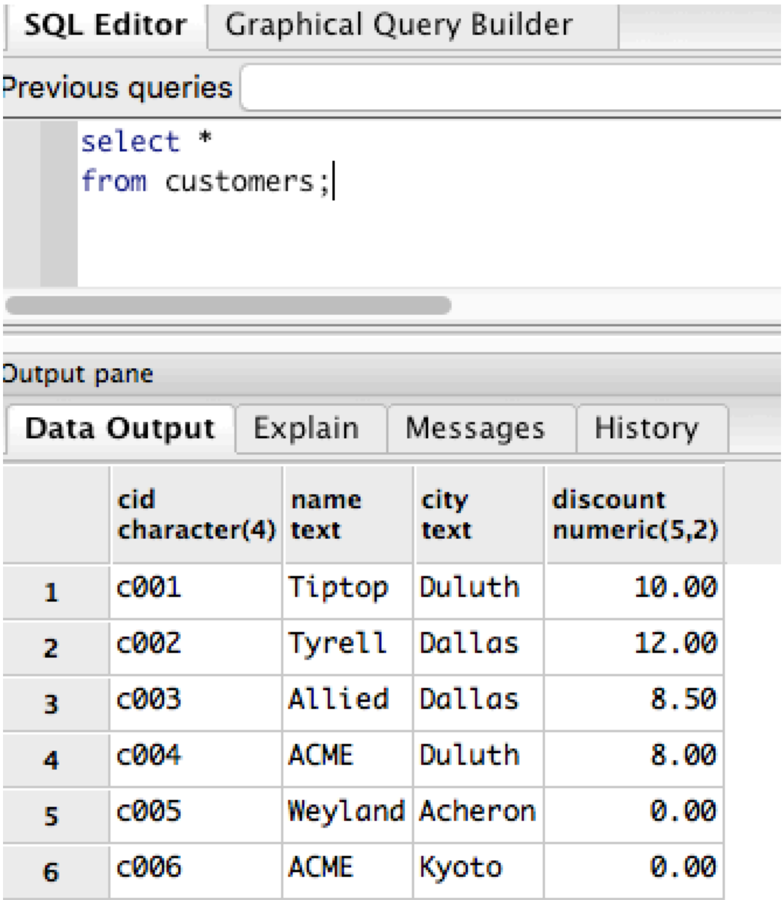
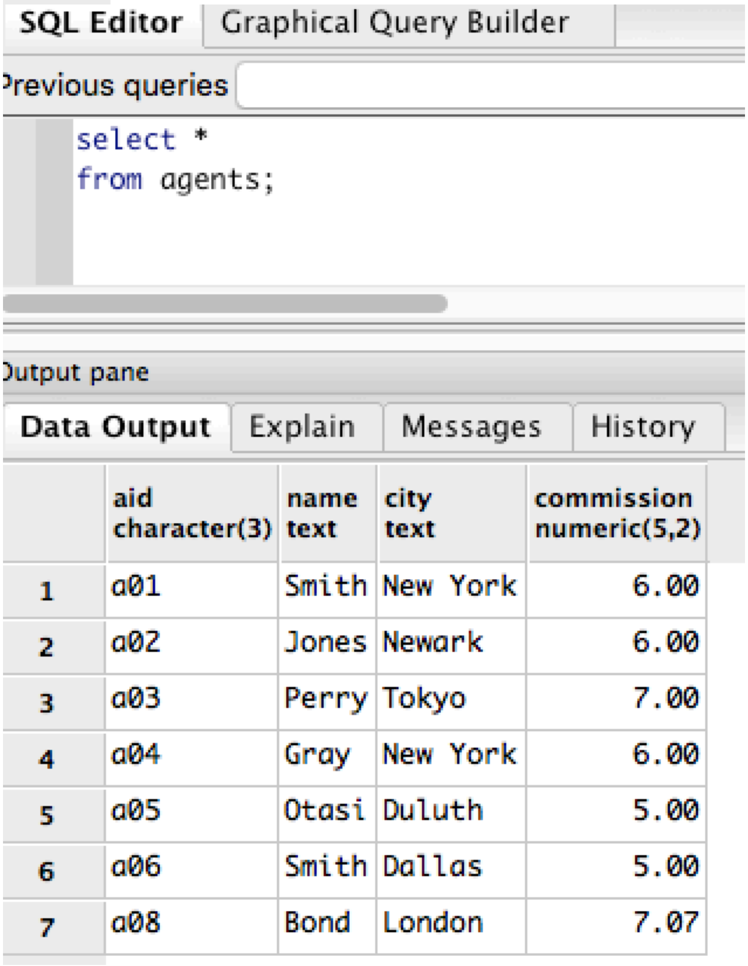


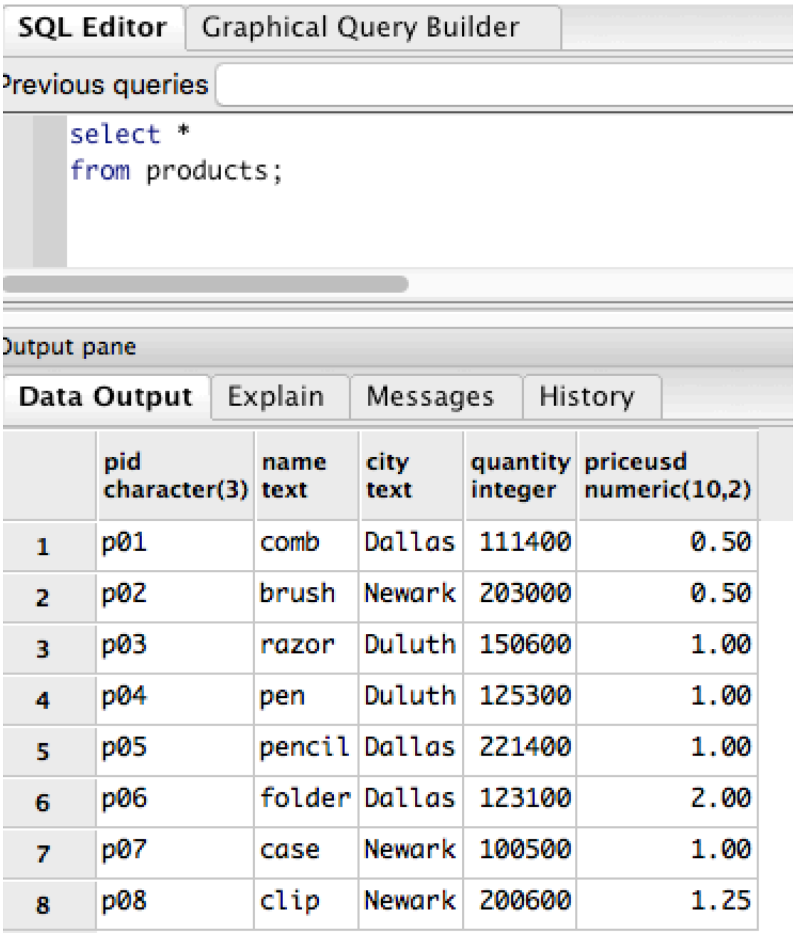
```sql
select *
from customers;
```

	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



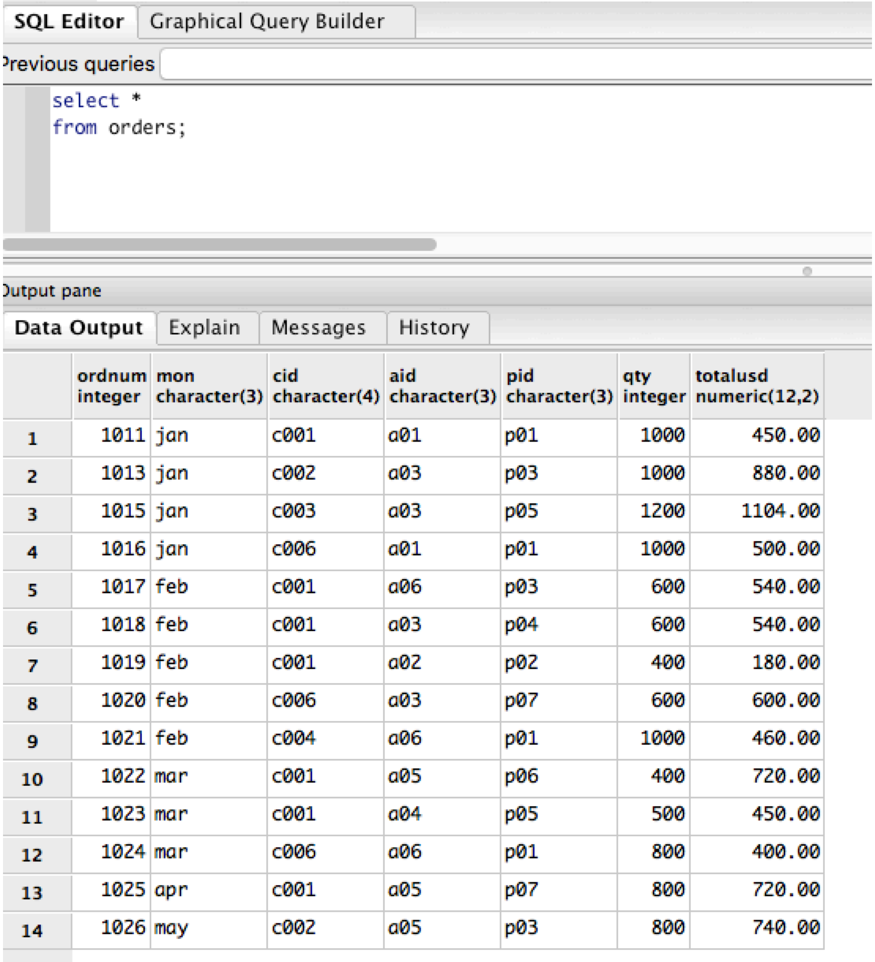
```sql
select *
from agents;
```

	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



```
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



```
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

2. A superkey is all the columns or set of columns that uniquely identifies any row in a column. Every key is a superkey but some superkeys are not minimal keys. A candidate key is a superkey with the fewest possible number of columns. A primary key is the candidate key that you choose to be primary. It uniquely identifies a row in a relation.

3. All attributes, which are listed in columns, must be given a data type. This is because the DBMS needs to know the kind of data that is working with. Marist College's database could have a table that contains information about students. Attributes of the students could be student ID, student name, GPA, major(s), minor(s) gender, and graduation date. Student name, major, and minor would both be the character string data type. The SQL declarations would be: studentName CHAR(50), major CHAR(100), and minor CHAR(100). The number in the parenthesis declares how many characters the string can contain. It's important not to limit the length of those fields so that's why I allowed for more characters than there would ever be needed. Using CHAR instead of VARCHAR means that SQL will give padding using blanks, to make n characters. Major could be nullable because some students don't declare a major for a while. GPA and student ID would both be declared as integers. The SQL declarations would be: studentID INT, and gpa INT. Unlike CHAR, SQL doesn't need you to declare the length. GPA could be nullable because there are some students who don't show up to class even though they are enrolled and they usually don't graded on a typical scale. Graduation data would use declared as the DATE data type. The SQL declaration would be: graduationDate DATE. When entering a date the form would be: '2018-05-16', the month and day get padded with a 0 if they are single digits. DATE values are actually characters that get converted to a special form using a function. DATE probably wouldn't be nullable because the graduation date could always be calculated and changed based on the number of credits a student has, but it won't ever be missing. The gender would be a character value type. The SQL declaration would be: gender CHAR DEFAULT '?'. Since I didn't declare the number of characters the value would be specified to 1, the default value. The second part of the declaration says that the gender could also be unknown because some people may not want to put down a gender. It seems better to set it to a default value instead of NULL.

4. a. The "first normal rule" is the condition that every component of every row is atomic. So if a there is a cell that has multiple values, then that database does not follow the rule. Shirts can have many colors, but instead of putting all the colors of a shirt in one cell of a table I can split them up into different rows. I could make a table that has the product ID and the price of the shirt as the attributes. Then I could make a table that has product ID and shirt color as the attributes. Now I'll have same colors and product ID's in the same columns, but I'll be abiding by the rule. It'll be much easier to search and filter for information.

b. There is no order to rows, which is the reason why we search for information by what value the row holds instead of row number. For example, we can't ask to look up information from a row 5 down from the top. The reason for this is that we probably won't know the information that a specific row holds, so it makes it much easier to be able to access content by what it is instead of where it is.

c. All rows must be unique with a table, so no two rows can have the same values in every column. Referring back to my example in a, even though the same values are repeated in the same column there won't be any duplicated rows. There a few cases where this rule isn't followed, but almost all the time this rule will be followed. If rows are the exact same then selecting the right one will bring up problems and no single column could be declared as a primary key.