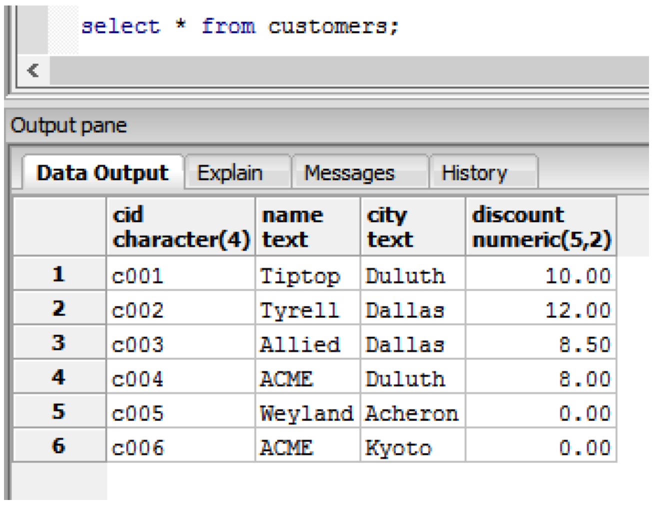
Krisztián Köves
CMPT 308

2/2/2016
Section 111

Lab 2 – CAP Database

1. Query screenshots:

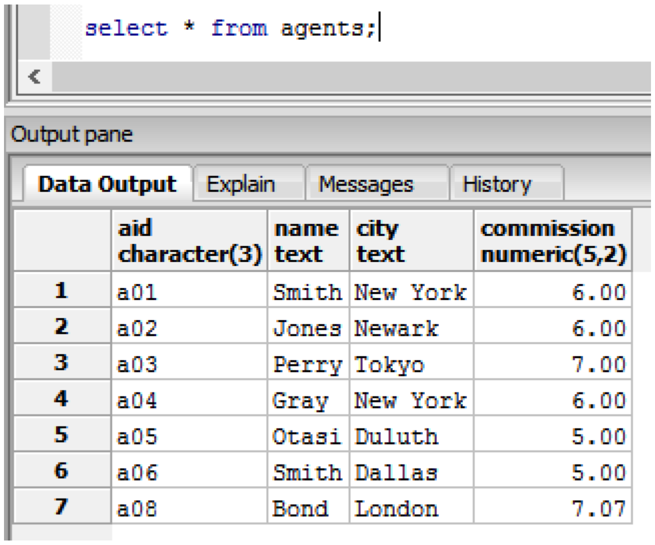


```
select * from customers;
```

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

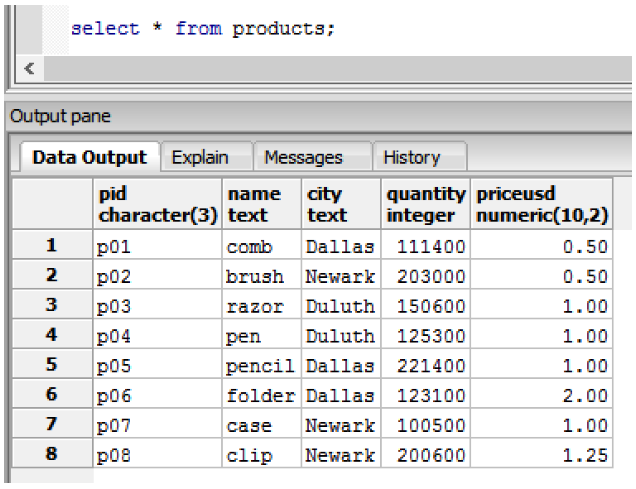


```
select * from agents;
```

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

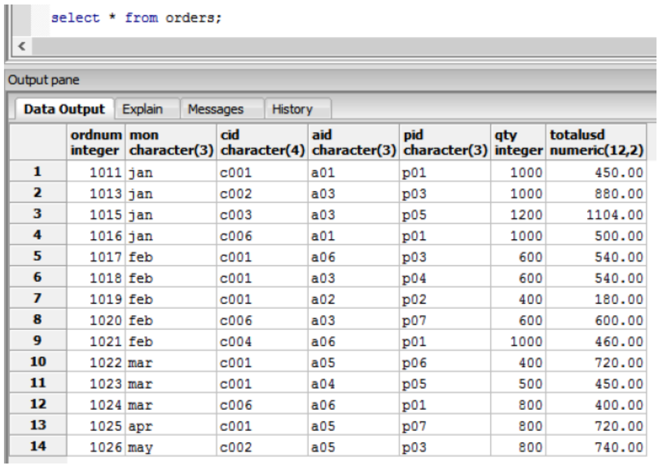


```
select * from products;
```

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



```
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

The query results are the same as the database snapshot in the `cap3.pdf` file.

2. A superkey is a set of columns in a database table that uniquely identifies all rows in the table. A candidate key is a minimal version of the superkey, with the least number of columns possible that still uniquely identifies rows in the table, and a primary key is generally the candidate key that we, the designers of the database, choose to use as the identifying column(s) for each row of the table.

3. My hypothetical database is a list of science fiction books, `SciFiBooks`. The columns would be as follows: `bookID`, `integer`, not nullable; `title`, `text`, not nullable; `author`, `text`, not nullable; `publishDate`, `date`, nullable (in the case that publish date is not available for some reason); and `avgRating`, `decimal`, nullable (in case a certain book has not been rated yet).

4. Explanation of relational rules:
    a. The first normal form rule tells us that a column in a relational database table cannot be made up of multiple attributes; they must be atomic data values, able to be represented by a single value in a single data type. For example, we could not have an object "person" – with multiple attributes – in a database column, since we can only store one value in each individual column. Thus, if we needed to store data about people in a table, rather than violating this law and storing several attributes in one column, we could make new columns for this data, or a new table altogether. This rule is important because it keeps our tables and data consistent – only one data value can be found at the intersection of any row and column.

    b. The "access rows by content only" rule states that we can only query for a database row by its content, not by its location. One example of this is that we could not query a database to give us row number 3 of a table, but we could ask it for the content in the row with an ID of 3. This way, if a row with a certain primary key was ever deleted, we will receive an empty response from the database, telling us that row no longer exists, whereas if we had been able to query for row number three, we would always get a response as long as the database is three or more rows long, giving us different data than we might have expected. This rule keeps our query results consistent, as we cannot ask for a row that does not exist.

    c. The "all rows must be unique" rule means just what the name implies: that all rows must be unique. This rule isn't as strongly enforced as the others – most database systems will not stop anyone from breaking this rule – because there are some cases in which rows would need to be not unique, such as in a linking table where there may be several entries for a single primary key. Nonetheless, this rule is important to keep data consistent, since duplicate entries could have errors that make the data invalid. This rule can be broken more easily than the others, but there should always be good reasoning – and comments in the code – to justify it.