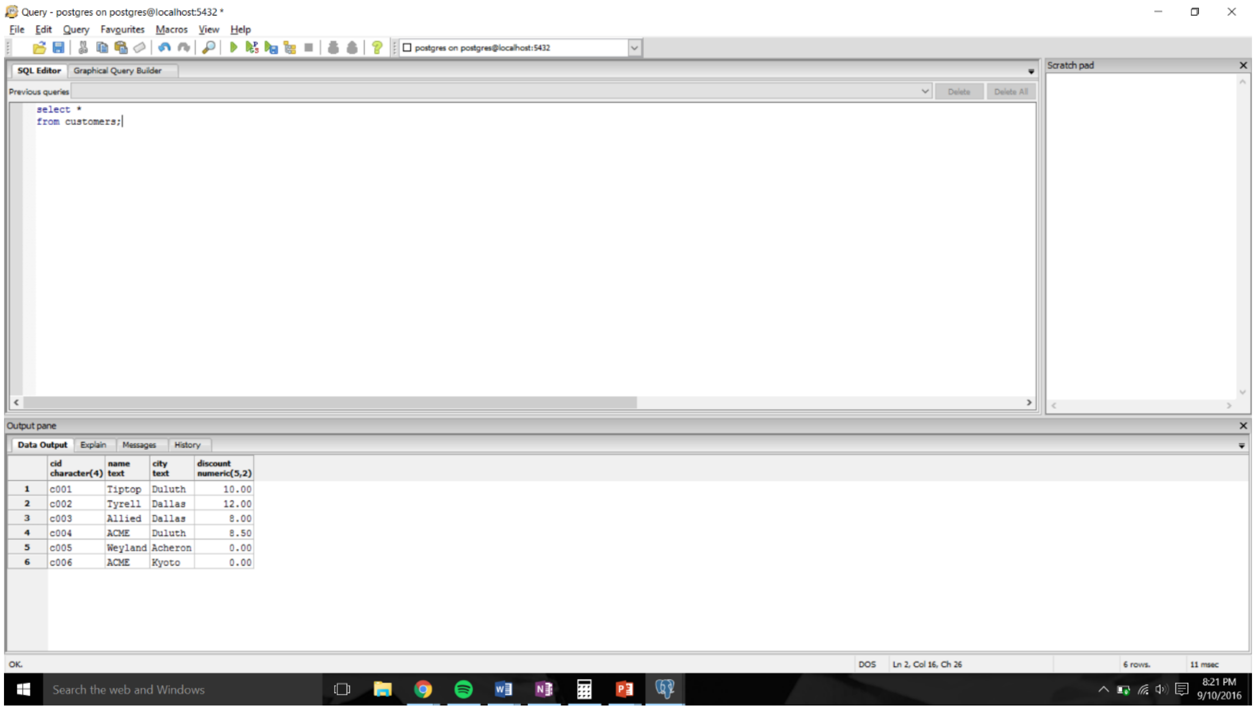
Tadd Bindas

9/10/16

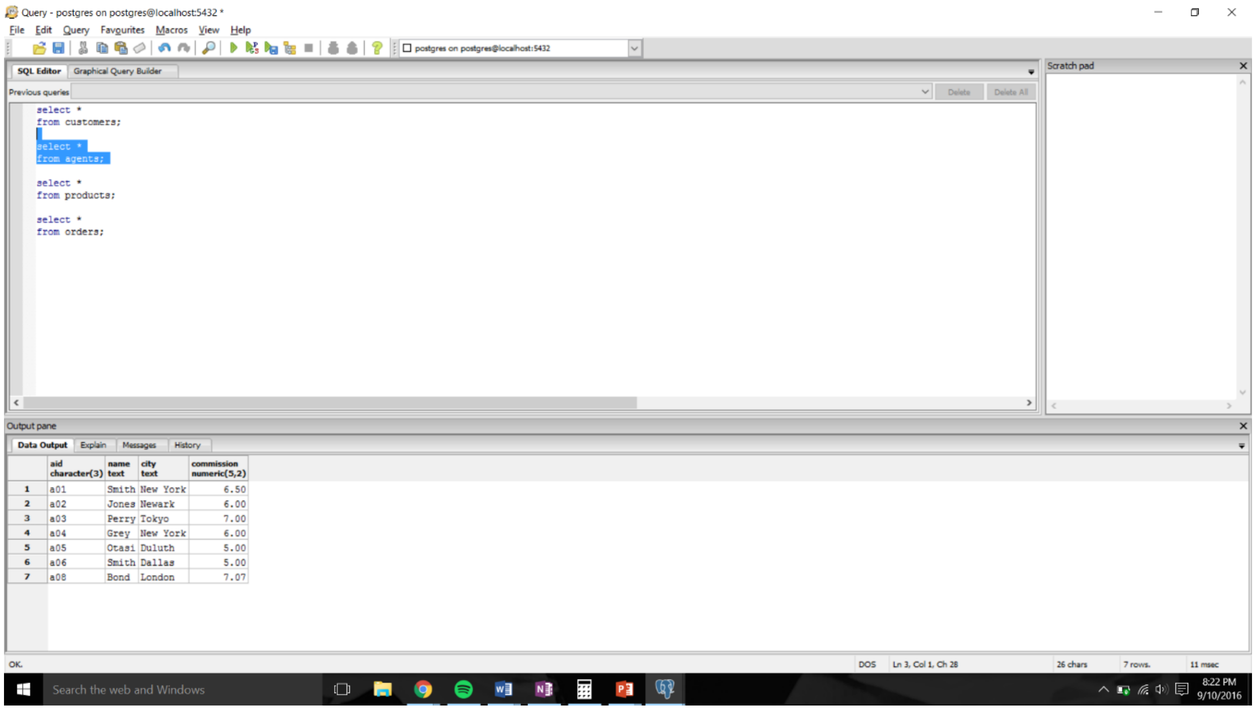
Lab-2


Query - postgres on postgres@localhost:5432 *

```
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	8.00
4	c004	ACME	Duluth	8.50
5	c005	Weyland	Acheron	0.00
6	c006	ACME	Kyoto	0.00

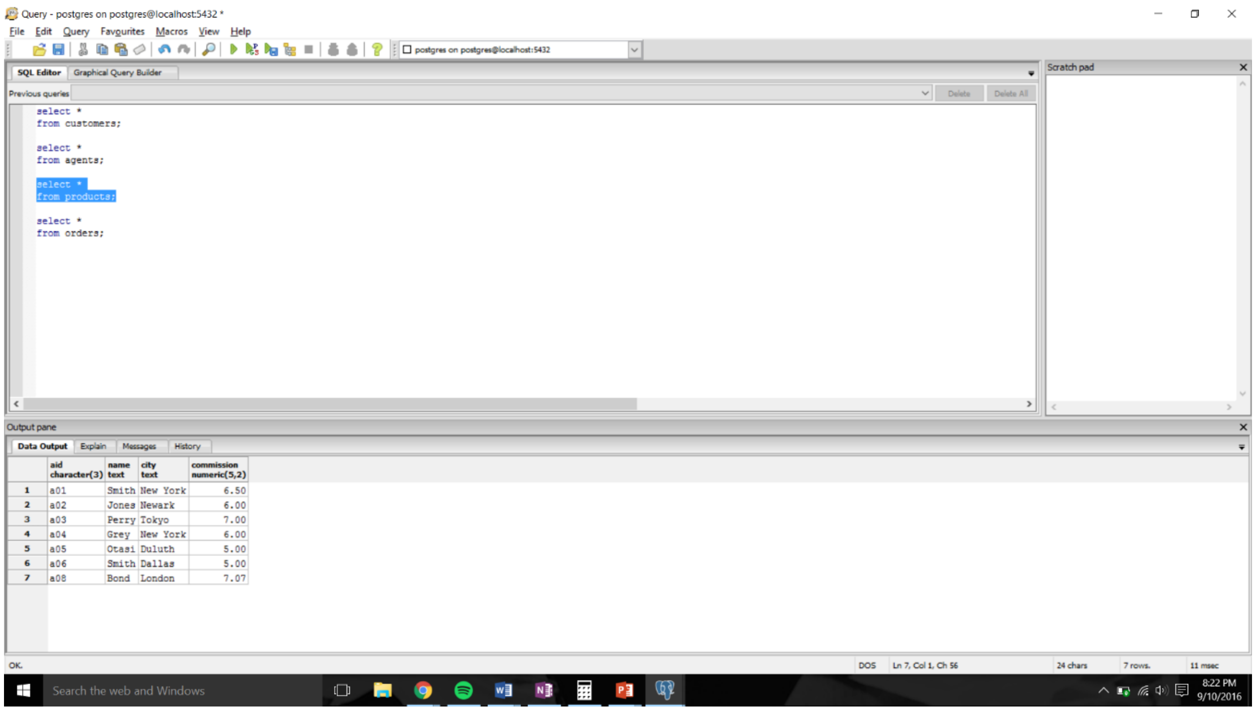
OK. DOS Ln 2, Col 16, Ch 26 6 rows. 11 msec


Query - postgres on postgres@localhost:5432 *

```
select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

	aid character(3)	name text	city text	commission numeric(5,2)
1	a01	Smith	New York	6.50
2	a02	Jones	Newark	6.00
3	a03	Perry	Tokyo	7.00
4	a04	Grey	New York	6.00
5	a05	Otasi	Duluth	5.00
6	a06	Smith	Dallas	5.00
7	a08	Bond	London	7.07

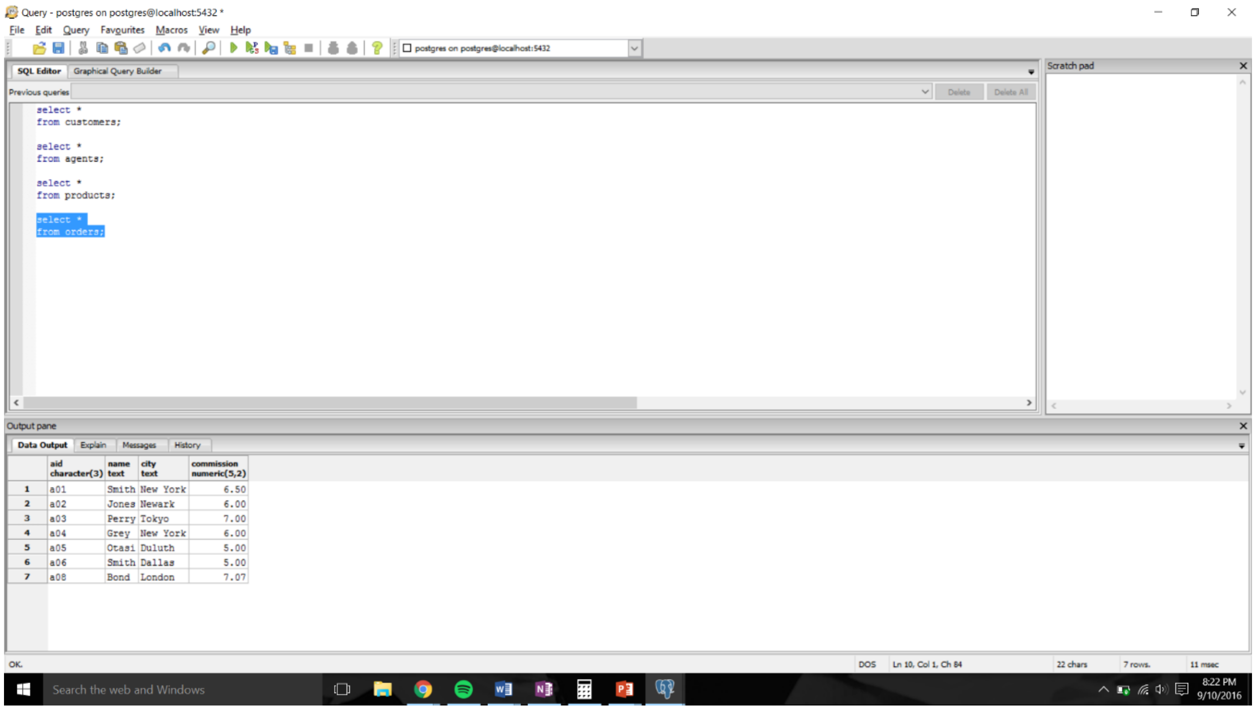
OK. DOS Ln 3, Col 1, Ch 28 26 chars 7 rows. 11 msec


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History
Search the web and Windows
8:22 PM
9/10/2016

```
select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

```
   aid           name   city      commission
   character(3)  text   text      numeric(5,2)
1  a01           Smith  New York          6.50
2  a02           Jones  Newark            6.00
3  a03           Perry  Tokyo             7.00
4  a04           Grey   New York          6.00
5  a05           Otasi  Duluth            5.00
6  a06           Smith  Dallas            5.00
7  a08           Bond   London            7.07
```

```
OK.    DOS    Ln 7, Col 1, Ch 56    24 chars    7 rows.    11 msec
```


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History
Search the web and Windows
8:22 PM
9/10/2016

```
select *
from customers;

select *
from agents;

select *
from products;

select *
from orders;
```

```
   aid           name   city      commission
   character(3)  text   text      numeric(5,2)
1  a01           Smith  New York          6.50
2  a02           Jones  Newark            6.00
3  a03           Perry  Tokyo             7.00
4  a04           Grey   New York          6.00
5  a05           Otasi  Duluth            5.00
6  a06           Smith  Dallas            5.00
7  a08           Bond   London            7.07
```

```
OK.    DOS    Ln 10, Col 1, Ch 84    22 chars    7 rows.    11 msec
```

I have the same results and data in my CAP database that the CAP3 Database has.

2) The terms primary key, candidate key, and super key are all different but related. A primary key is a special relational database column that is used to uniquely identify each record. The primary key cannot contain null values and each primary key must be unique. In the CAP database, the aid, pid, and oid are all example of primary keys. A candidate key is a set of columns, or a single column, that can identify any database record without any additional data. A primary key is a special version of a candidate key, and each table can have more than one candidate key. A super key is the combination of candidate keys that are used to uniquely identify each value in the database. All of these keys are all related to another but all different.

3) A data type is a specific format that data is in and specific operations that can be performed. There are many data types in PostgreSql. Boolean, text, var char, and date to name a specific few. The operations that can be performed on each data type vary per type. For example, you can add, subtract, and multiply ints. You cannot, however, muliplty a text or var char. My imaginary topic that I am going to make a table for is "Food in the refrigerator at my house." Inside the table there will be the following rows

FID	Name	Food _Group	Meal	Date_Bought	Exipration_Date

The row FID is the primary key of the entire table. It will be an int and will be a not null unique value for each item. The name will be a var char with a limit of 64 characters. This will be the name of the item and will be a non-null value. The Food_Group value will be a varchar value character limit 16. This group will list the food group of the item as a way of classifying the items. The meal row classifies the items in what meal of the day they belong to, or if they are a snack. It is a var char. The Date_bought and Expiration_Date fields will be date fields. It will correspond to the date the item was bought and the date it expires. Out of all of the fields, only Meal and Food_Group can be null, the rest are non-null values.

4)a) The first normal form rule is that all fields are atomic. What this means is that all fields are the most simplified version of the data to prevent any confusion. For example, in the table that I created above in question 3, the field name is atomic because you cannot simplify the name of a food item down any farther. If the column "name" were in a table of people, the column would not be atomic because you could simplify it to first name and last name, breaking rule #1.

b) The second normal form rule is "Access by what, never where." This rule means that you cannot access data based on where the data is on the table, only sing a select statement to try to gather data for what you need. An example is the game battleship. You should not access data like you play battleship (calling coordinates of locations). You need to use a select statement to determine the data you need.

c) The third row is that all rows are unique. This row means simply that no two rows can be the same. A way to follow this rule is to have a primary key that is unique. Even with every item in my food table containing the same name, food group, meal, date bought, and expiration date, the primary key will help distinguish the items and make all rows unique.