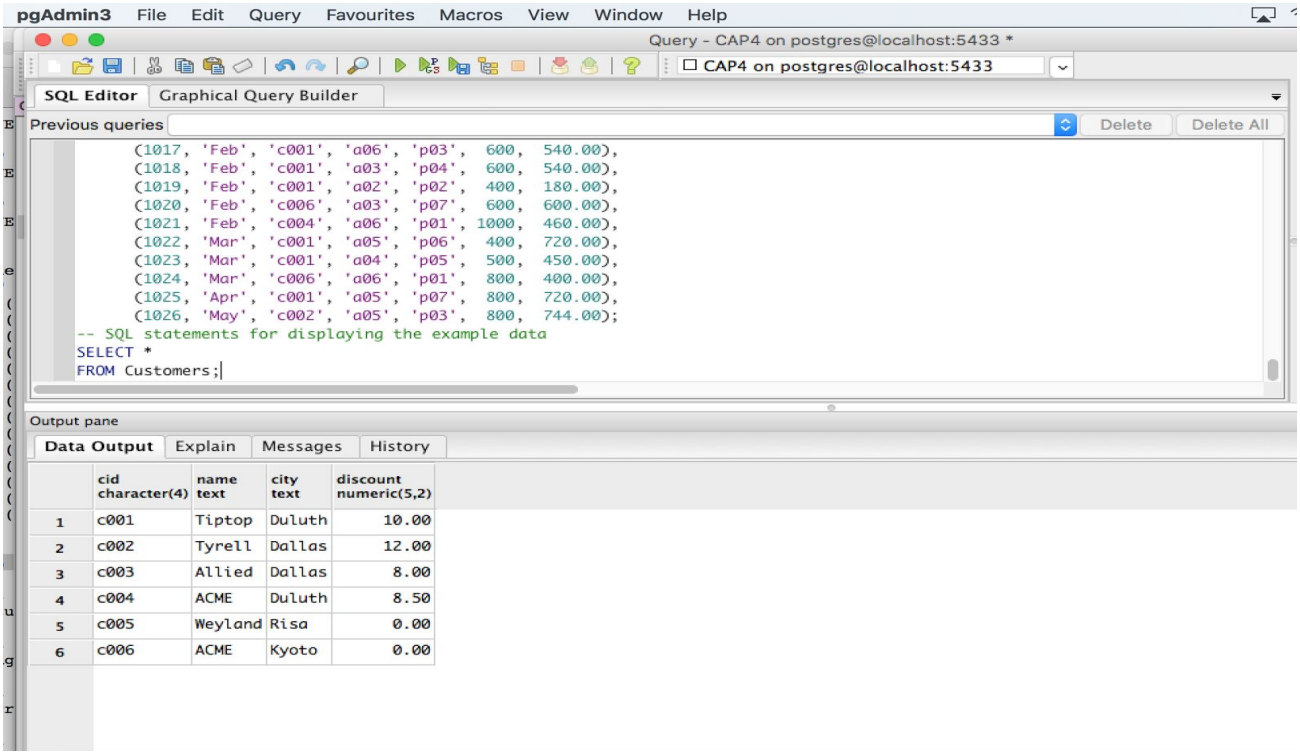
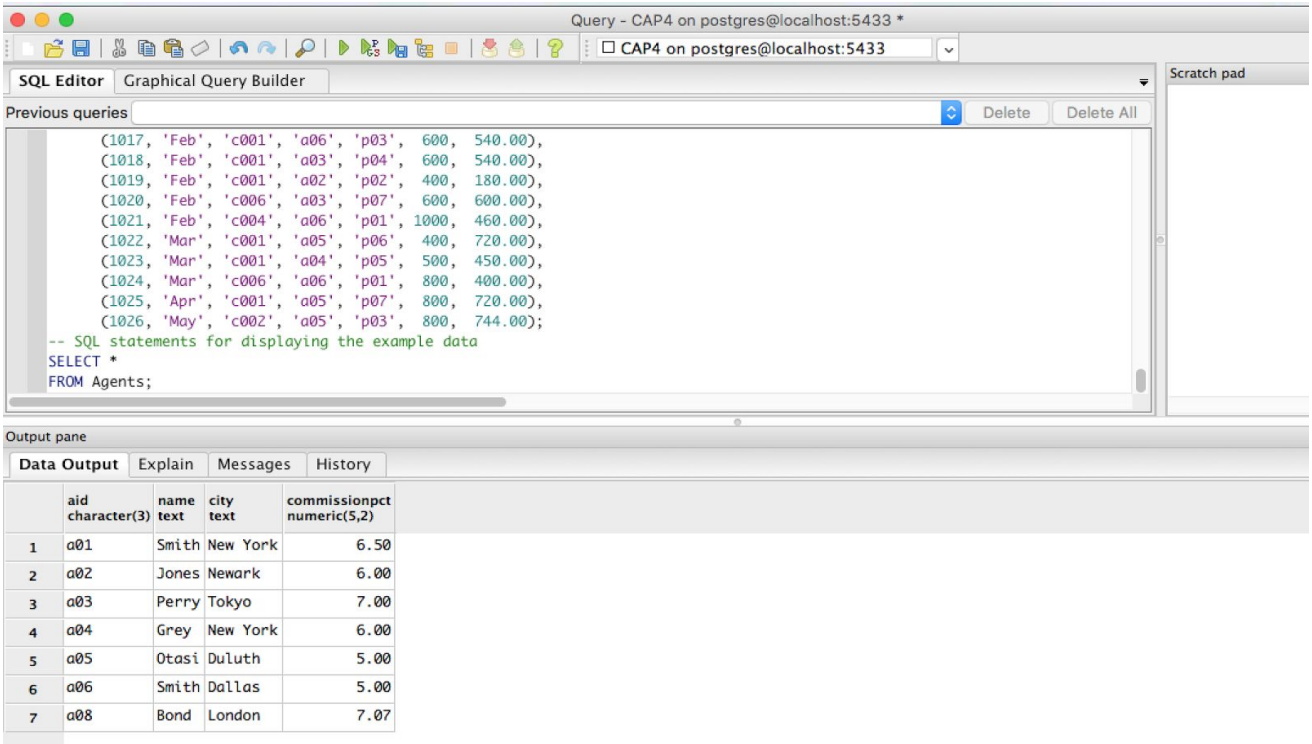
Simoni Handoo
01/28/2017
Lab 2

Section 1:

SELECT *
FROM Customers;

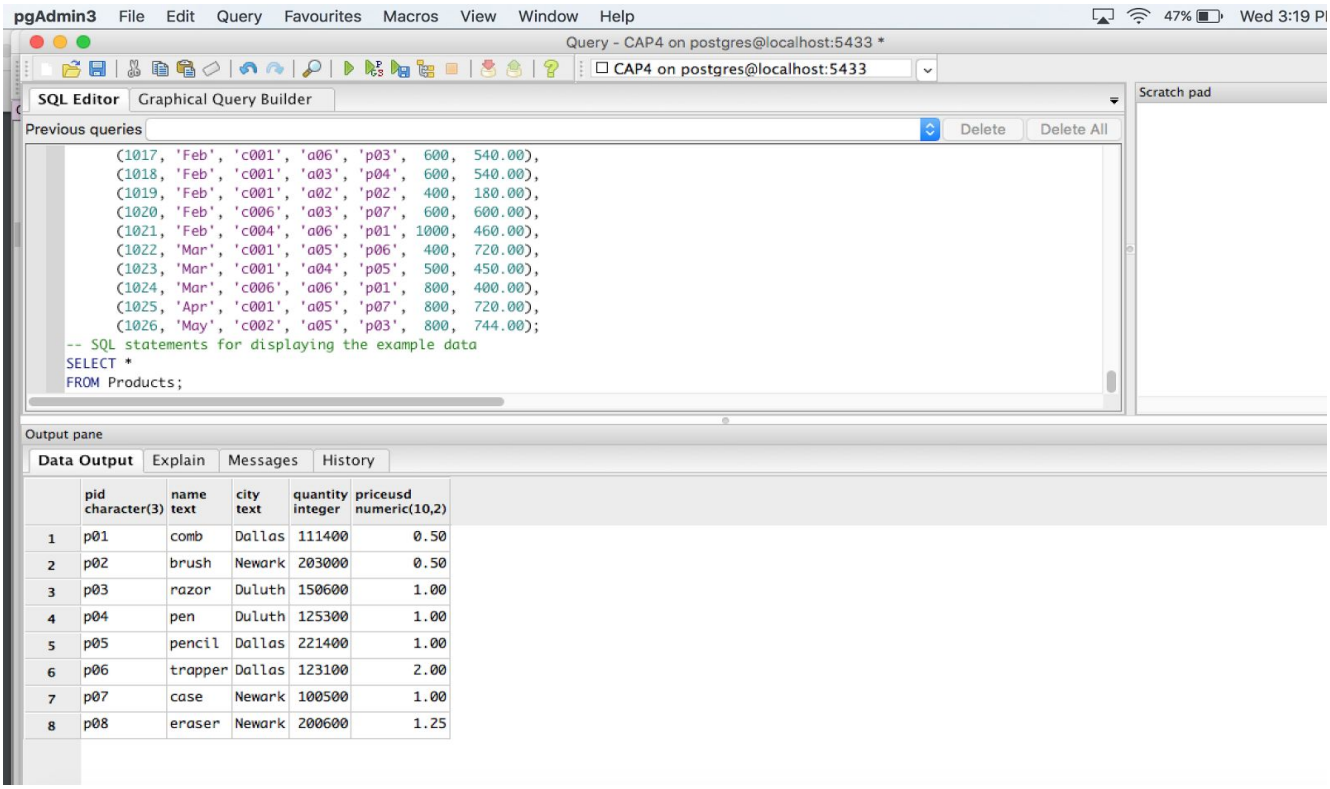


SELECT *
FROM Agents;



Simoni Handoo
01/28/2017
Lab 2

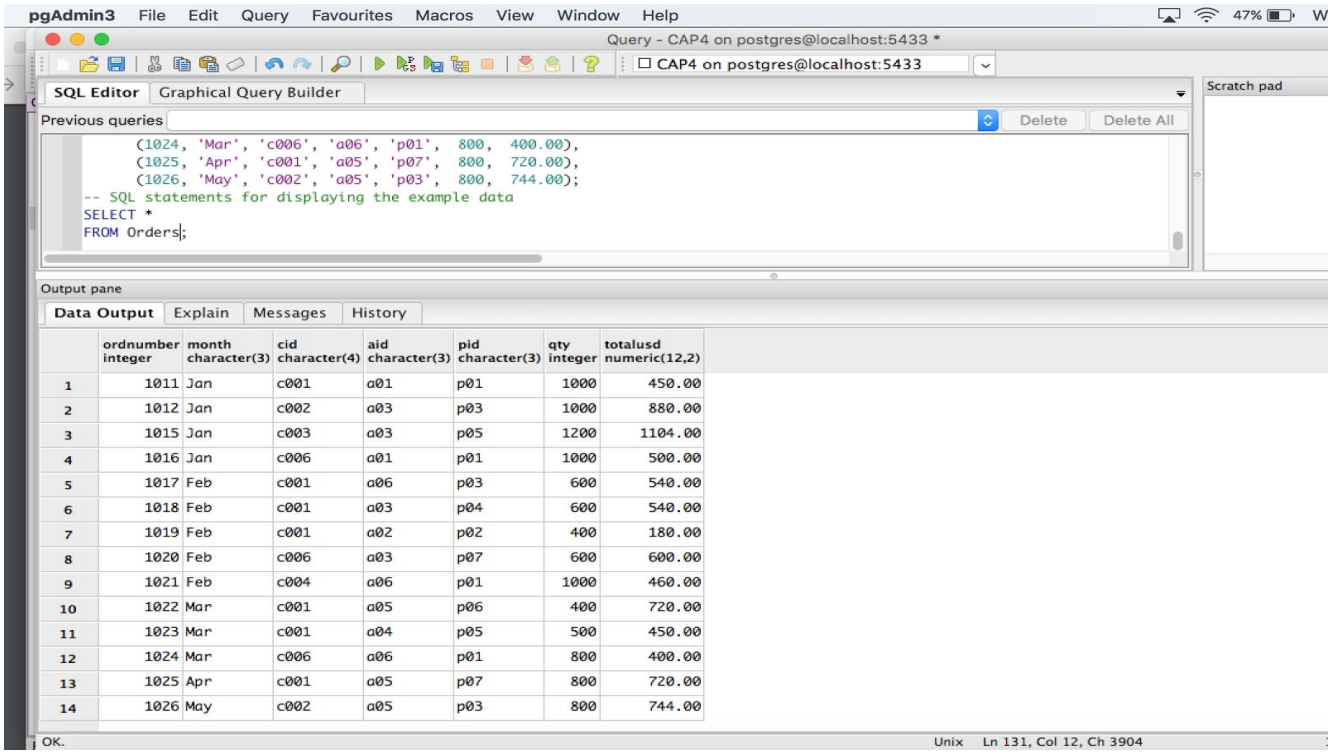
**SELECT *
FROM Products;**



```
(1017, 'Feb', 'c001', 'a06', 'p03',  600,  540.00),
(1018, 'Feb', 'c001', 'a03', 'p04',  600,  540.00),
(1019, 'Feb', 'c001', 'a02', 'p02',  400,  180.00),
(1020, 'Feb', 'c006', 'a03', 'p07',  600,  600.00),
(1021, 'Feb', 'c004', 'a06', 'p01', 1000,  460.00),
(1022, 'Mar', 'c001', 'a05', 'p06',  400,  720.00),
(1023, 'Mar', 'c001', 'a04', 'p05',  500,  450.00),
(1024, 'Mar', 'c006', 'a06', 'p01',  800,  400.00),
(1025, 'Apr', 'c001', 'a05', 'p07',  800,  720.00),
(1026, 'May', 'c002', 'a05', 'p03',  800,  744.00);
-- SQL statements for displaying the example data
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	trapper	Dallas	123100	2.00
7	p07	case	Newark	100500	1.00
8	p08	eraser	Newark	200600	1.25

**SELECT *
FROM Orders;**



```
(1024, 'Mar', 'c006', 'a06', 'p01',  800,  400.00),
(1025, 'Apr', 'c001', 'a05', 'p07',  800,  720.00),
(1026, 'May', 'c002', 'a05', 'p03',  800,  744.00);
-- SQL statements for displaying the example data
SELECT *
FROM Orders;
```

	ordnumber integer	month 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	1012	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	744.00

The SQL queries executed above display the same data that is entered in the CAP4 snapshot of each table Customer, Products, Agents and Orders.

Section 2:

Primary Key	**Candidate Key**	**Superkey**
A primary key is a minimal candidate key with no NULL values. There is no difference between a primary key and any other candidate key.	A Candidate key can be seen as a "minimal superkey"- meaning the smallest subset of super key attributes which are unique.	A Superkey is any set of attributes for which the values are guaranteed to be unique for all possible sets of tuples in a table at ALL TIMES

Section 3: Data Types

In SQL, datatype is an attribute that specifies the type of data of any object. For example, if you are entering a numeric value, that field needs to have a data type indicating what type of numeric is being entered. In a SQL table, each column and/or variable has a data type. Data types are always used while creating a table in the database.

The SQL server offers 6 categories of data types:

1. Exact numeric data type (e.g. int, decimal, money etc.)
2. Approximate numeric data type (e.g. float)
3. Date and Time data type (e.g. datetime)
4. Character String data type (e.g. char, varchar, text etc.)
5. Unicode Character Strings data type (e.g. nchar, ntext etc.)
6. Binary data type (e.g. binary, image etc.)
7. Miscellaneous data type (e.g. timestamp)

Database: Marist College's students' records over the years
Table: Student's personal information

```
CREATE table IF NOT EXISTS MaristStudents
(
StudentID      INT UNIQUE   NOT NULL   AUTO_INCREMENT,
LastName       TEXT   NOT NULL,
FirstName      TEXT,
Address        TEXT,
PhoneNumber INT
);
```

The StudentID column and the LastName column do not accept NULL values, meaning that these fields (columns) cannot have NO value- they need to be assigned some kind of value otherwise it will give an error. The UNIQUE constraint uniquely identifies each record in a database table. The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns. AUTO_INCREMENT is available only for numeric

columns, to automatically generate a number that is one more than the previous value in that column. TEXT specifies that the values entered are in characters.

Section 4: Relational “Rules”

1. “First normal form” rule: sets the basic rules for an organized database
    a. First rule of 1NF: You must define the data items. This is important as this looks at the data to be stored, organizes data into columns and defines what type of data each column contains. For example, all the columns in a business enterprise database relating to location of meetings should be kept in the Locations table, and those relating to customers in the Customers table
    b. Second rule of 1NF: The 2nd rule is ensuring that there are no repeating groups of data. This is important as we should always aim to reduce redundancy

Example:

Student ID	Name	Phone number
20067792	John	8458293749 3287489127
20069383	Mary	9287380298 2632879814
20063283	Alicia	6782637280 5912739237
20078362	Craig	8457299273 7123987183

In the case above, we have to phone numbers for each student in the same column. For this table to follow the first normal form rule, it should have a separate column for home phone number and cell phone number.

Student ID	Name	Home phone number	Cell phone number
20067792	John	8458293749	3287489127
20069383	Mary	9287380298	2632879814
20063283	Alicia	6782637280	5912739237
20078362	Craig	8457299273	7123987183

Now, the table above follows the First Normal Form rule, that there are no repeating groups of data.

   c. Third rule of 1NF: Create a primary key for each table which we have already created. So for example, the StudentID field (column) of a student attending Marist College could be a Primary key as it is unique. This is important as having a primary key ensures row-level accessibility. It makes it convenient for a user to add, sort, modify, or delete data in a database.
2. “Access rows by content only” rule: Accessing rows by content i.e. accessing rows by the values of their columns. You cannot access rows by pointer or the number of rows. An example from the CAP4 database would be accessing “OrdNumber 1015” instead of accessing the row by saying “Row 5”. You cannot just retrieve the fourth row from the ORDERS table from CAP4 Database. Instead, the query has to refer to the row with the value of 1019 from the ordno column which is a unique row. It’s about the “WHAT” (content), not “WHERE.”
3. “All rows must be unique” rule: You cannot have duplicate tuples. In other words, two tuples in a relation can not be identical in all column values at once. We can think of relation as a set of tuples that contains no two similar elements. Each tuple of the set is unique. For example, in commercial database systems, they insure that newly inserted row in a table for instance doesn’t duplicate an existing row, which is a very important aspect of commercial database.