```
147  select *
148  from Customers;
149
```

Data Output | Explain | Messages | History

cid character	name text	city text	discount numeric ...
c001	Tiptop	Duluth	10
c002	Tyrell	Dallas	12
c003	Allied	Dallas	8
c004	ACME	Duluth	8.5
c005	Weyland	Risa	0
c006	ACME	Kyoto	0

```
150  select *
151  from Agents;
152
```

Data Output | Explain | Messages | History

aid character	name text	city text	commissi... numeric ...
a01	Smith	New York	6.5
a02	Jones	Newark	6
a03	Perry	Tokyo	7
a04	Grey	New York	6
a05	Otasi	Duluth	5
a06	Smith	Dallas	5
a08	Bond	London	7.07

```
152
153  select *
154  from Products;
```

Data Output | Explain | Messages | History

pid character	name text	city text	quantity integer	priceusd numeric ...
p01	comb	Dallas	111400	0.5
p02	brush	Newark	203000	0.5
p03	razor	Duluth	150600	1
p04	pen	Duluth	125300	1
p05	pencil	Dallas	221400	1
p06	trapper	Dallas	123100	2
p07	case	Newark	100500	1
p08	eraser	Newark	200600	1.25

```
155
156 select *
157 from Orders;
```

Data Output | Explain | Messages | History

ordnumb... integer	month character	cid character	aid character	pid character	qty integer	totalusd numeric ...
1011	Jan	c001	a01	p01	1000	450
1012	Jan	c002	a03	p03	1000	880
1015	Jan	c003	a03	p05	1200	1104
1016	Jan	c006	a01	p01	1000	500
1017	Feb	c001	a06	p03	600	540
1018	Feb	c001	a03	p04	600	540
1019	Feb	c001	a02	p02	400	180
1020	Feb	c006	a03	p07	600	600
1021	Feb	c004	a06	p01	1000	460
1022	Mar	c001	a05	p06	400	720
1023	Mar	c001	a04	p05	500	450
1024	Mar	c006	a06	p01	800	400
1025	Apr	c001	a05	p07	800	720
1026	May	c002	a05	p03	800	744

Michael Hercules Sirico
1/31/2017
Lab2
Database Management

Upon Comparison of all of the above screenshots to the image of data tables provided, There appears to be no discrepancies.

Differentiate between Primary, Candidate, and Super Keys

A Primary Key is a column in which each row has a unique value that can be used to Identify that particular row.

A Candidate Key is an individual column in a table that qualifies to be a Primary Key; that is, all values in the column are unique to each Row.

A Super Key is a Primary Key in addition to another column. The addition of a Primary Key with another column creates a Super Key.

Data Types

In choosing a topic to create a table, I decided to select Coffee as a topic. The Name of the table is Coffee. Our Coffee Table has multiple columns, or “fields.” The fields are as follows: “Coffee Id”, “Name of Blend”, “Has Milk”, “Amount of Milk in Ounces”, “Type of Milk”, “Has Sugar”, “Amount of Sugar in teaspoons”, “Type of Sugar”, “Price of Coffee in USD”. The data type for each field will depend on what it needs to hold; “Coffee Id” will be an Integer, “Name of Blend” will be a String, “Has Milk” will be a Boolean, “Amount of Milk in Ounces” will be an Integer, “Type of Milk” will be a String, “Has Sugar” Will be a Boolean, “Amount of Sugar in teaspoons” Will be an Integer, “Type of Sugar” will be a String, and “Price of Coffee in USD” will be an Integer. Despite my learning that Postgresql has a currency, or “Money” Data Type, it appears that using it may not be the best choice in this implementation. Lastly, in Regards to Nullability; Certain Fields will need to be able to be Null, While others should Not Be able to be Null. “Coffee Id” cannot be null, “Name of Blend” cannot be null, “Has Milk” cannot be null, “Amount of Milk in Ounces” can be null, “Type of Milk” can be null, “Has Sugar” cannot be null, “Amount of Sugar in teaspoons” can be null, “Type of Sugar” can be null, “Price of Coffee in USD” cannot be null. Following this, there is a table that describes all of the fields that will be incorporated into our coffee table.

“Coffee Table” description table, from Data Types Short Essay

Name of Field in Coffee Table	Data Type of Field in Coffee Table	Nullability Field in Coffee Table
Coffee Id	Integer	Not Nullable (Can not be Null)
Name of Blend	String	Not Nullable (Can not be Null)
Has Milk	Boolean	Not Nullable (Can not be Null)
Amount of Milk in Ounces	Integer	Nullable (Can be Null)
Type of Milk	String	Nullable (Can be Null)
Has Sugar	Boolean	Not Nullable (Can not be Null)
Amount of Sugar in Teaspoons	Integer	Nullable (Can be Null)
Type of Sugar	String	Nullable (Can be Null)
Price of Coffee in USD	Integer	Not Nullable (Can not be Null)

Relational Database ‘Rules”

A. The First Normal Form Rule is important to ensure that data is not unnecessarily duplicated; each field refers to only a single thing, not multiple things; which prevents confusion over which, in addition to unnecessarily duplicating data that Is already stored. An example could be in a table of possessions; if one of the columns is “Cars” but a person owns multiple cars, rather than having a set of cars within a single field, it spreads them out so that there is no smaller sets within the field.

B. The Rule that states you must “access rows by content only” is extremely important because of how tables work. When querying a table, rows can shift location, be returned in an order they are not stored in, and so on. So to ensure that the row accessed is the intended row, accessing it by its content is a way to guarantee the row accessed is the row intended. An example is a table with 3 rows. If at first, you try to access row 2, and it brings up the row. Later on, row 1 is deleted. Then what was row 2, has now become row 1. Accessing a row by content eliminates any need or dependence on order of the rows within the table.

C. “All Rows must be unique.” If two rows exist that are exactly the same, there’s no way to differentiate or distinguish them. It creates redundancy as well as conflicts with the data, which can lead to large scale confusion. An example of this, and how to solve it, is if in a table of Cars, there are two “Red” “2003” “Honda” “Civic” entries. Two of them exist, but if the rows are identical, there is no way to distinguish them, and the data is redundant. To solve this, setting, or creating a primary key; an ID column is the answer; a unique column so each row has at least one unique field.