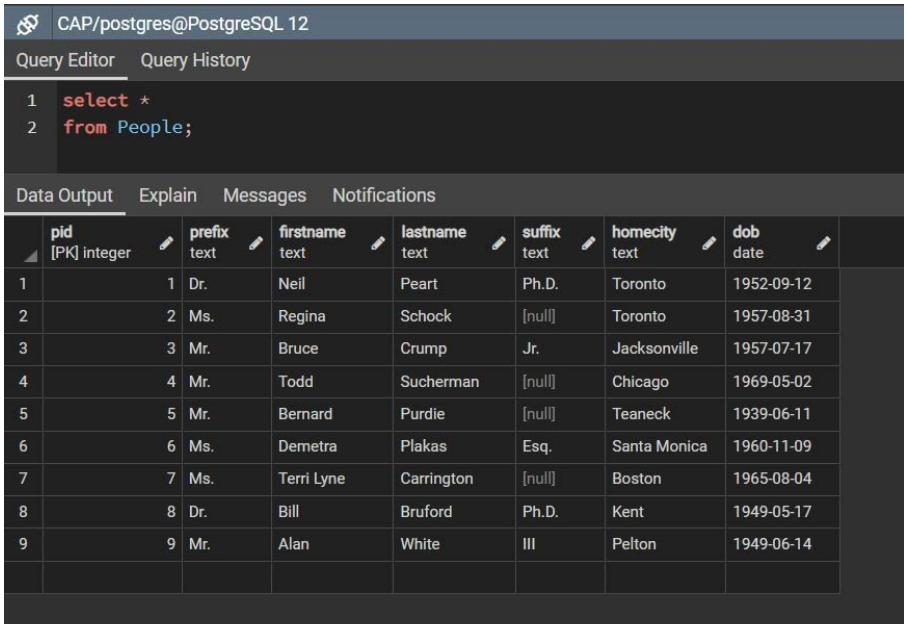
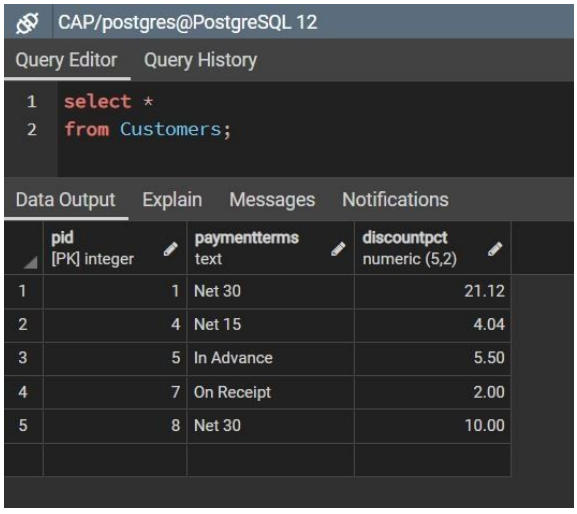
Michael Wise
Dr. Alan Labouseur
CMPT 308 Database Management
10 September 2020

Lab #2: CAP Database

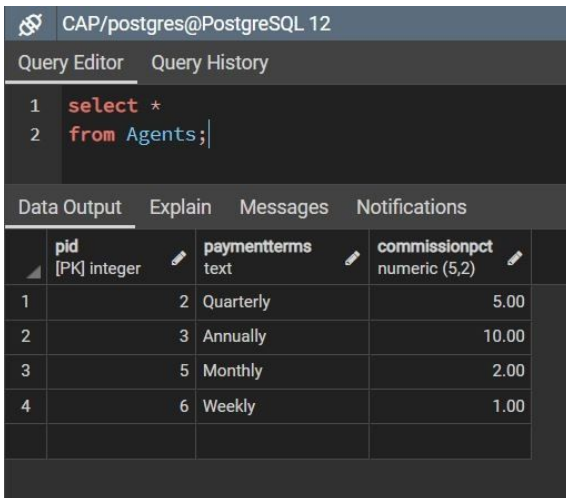
1) Queries for our CAP Database:



	pid [PK] integer	prefix text	firstname text	lastname text	suffix text	homecity text	dob date
1	1	Dr.	Neil	Peart	Ph.D.	Toronto	1952-09-12
2	2	Ms.	Regina	Schock	[null]	Toronto	1957-08-31
3	3	Mr.	Bruce	Crump	Jr.	Jacksonville	1957-07-17
4	4	Mr.	Todd	Sucherman	[null]	Chicago	1969-05-02
5	5	Mr.	Bernard	Purdie	[null]	Teaneck	1939-06-11
6	6	Ms.	Demetra	Plakas	Esq.	Santa Monica	1960-11-09
7	7	Ms.	Terri Lyne	Carrington	[null]	Boston	1965-08-04
8	8	Dr.	Bill	Bruford	Ph.D.	Kent	1949-05-17
9	9	Mr.	Alan	White	III	Pelton	1949-06-14



	pid [PK] integer	paymentterms text	discountpct numeric (5,2)
1	1	Net 30	21.12
2	4	Net 15	4.04
3	5	In Advance	5.50
4	7	On Receipt	2.00
5	8	Net 30	10.00



	pid [PK] integer	paymentterms text	commissionpct numeric (5,2)
1	2	Quarterly	5.00
2	3	Annually	10.00
3	5	Monthly	2.00
4	6	Weekly	1.00

CAP/postgres@PostgreSQL 12

Query Editor | Query History

```
1  select *
2  from Products;
```

Data Output | Explain | Messages | Notifications

	prodid [PK] character (3)	name text	city text	qtyonhand integer	priceusd numeric (10,2)
1	p01	Heisenberg Compensator	Dallas	47	67.50
2	p02	Universal Translator	Newark	2399	5.50
3	p03	Commodore PET	Duluth	1979	65.02
4	p04	LCARS module	Duluth	3	47.00
5	p05	Remo drumhead	Dallas	8675309	16.61
6	p06	Trapper Keeper	Dallas	1982	2.00
7	p07	Flux Capacitor	Newark	1007	1.00
8	p08	HAL 9000 memory core	Newark	200	1.25
9	p09	Red Barchetta	Toronto	1	379000.47

CAP/postgres@PostgreSQL 12

Query Editor | Query History

```
1  select *
2  from Orders;
```

Data Output | Explain | Messages | Notifications

	ordernum [PK] integer	dateordered date	custid integer	agentid integer	prodid character (3)	quantityordered integer	totalusd numeric (12,2)
1	1011	2020-01-23	1	2	p01	1100	58568.40
2	1012	2020-01-23	4	3	p03	1200	74871.83
3	1015	2020-01-23	5	3	p05	1000	15696.45
4	1016	2020-01-23	8	3	p01	1000	60750.00
5	1017	2020-02-14	1	3	p03	500	25643.88
6	1018	2020-02-14	1	3	p04	600	22244.16
7	1019	2020-02-14	1	2	p02	400	1735.36
8	1020	2020-02-14	4	5	p07	600	575.76
9	1021	2020-02-14	4	5	p01	1000	64773.00
10	1022	2020-03-15	1	3	p06	450	709.92
11	1023	2020-03-15	1	2	p05	500	6550.98
12	1024	2020-03-15	5	2	p01	880	56133.00
13	1025	2020-04-01	8	3	p07	888	799.20
14	1026	2020-05-01	8	5	p03	808	47282.54

2) In a table, a super key is just any combination of columns that uniquely identify every row in a table. A candidate key is best described as a field/column (a "minimal superkey") in which every value is unique from one another. A good example would be the CWID numbers in Marist's database. Each student has a unique ID, and we can use that to distinguish each specific student because of the fact that no two students share the same one. Simply put, a primary key is just one of the candidate keys that we choose to

uniquely identify a record. There can be multiple candidate keys, but ultimately only one primary key. I think Marist would rather choose CWID as the primary key over something like students' social security numbers (a more sensitive and secure value, yet still technically a candidate key).

3) Every table can have a variety of different data types. For example, let's make a table that identifies statistics for NBA players. The fields can be pid, firstName, lastName, suffix, dateOfBirth, position, team, and PPG. The first field, pid, will be an integer that represents a unique ID every individual player has. It is not nullable, as every player needs to have one. (This should also be the primary key). The next two fields, firstName & lastName, would be text that lists the first and last name of each player. They would not be nullable. However, the next field, suffix (text), is nullable as some players will have suffixes (Jr., Sr., III, etc.) while others will not. Assuming we are using postgreSQL, the next field, dateOfBirth, would be of type date, and is not nullable. Similar to first/last name, the field position (PG,SG,SF,PF,C) and team (Bucks, Lakers, Celtics, etc.) are both text. Every player needs a listed position (not nullable) but players can be free agents and technically not be assigned to a team (nullable). The PPG will be a decimal, as it will contain the values of every player's career average points per game (not nullable). You could go on and on with different statistical categories for players but the main point is that many different data types are used.

4) Analyzing the Relational Rules
    a) **The "first normal form" rule:** at any intersection of a row and column, each attribute can only contain a single value. That value has to be atomic, meaning they cannot be subdivided any further. This rule is important to eliminate redundancy and integrity within our data. For example, if we had a field named phoneNumbers and the entry read "555-123-4567, 777-135-246", we could still split it down further into columns (say cellNumbers and workNumbers).
    b) **The "access rows by content only" rule:** this rule says that we can only query data by what is there, and never by where it is. Thus, we can ask, "What is the address of the student with CWID 20123456?", but not something like "What is the birthday listed in the 3rd row?". This is important because there is no guarantee that tables will have the same order every time. Since tables are represented by sets, the order of the elements in them is not important. {a,b,c} = {b,c,a}
    c) **The "all rows must be unique" rule:** all rows must be distinct (hence no duplicates). This is crucial because of the same reason described in b). Tables have no intrinsic order, and if one were able to have two identical rows, you would have no way to tell them apart from one another. Also it just looks stupid to have to rows with the EXACT same information.