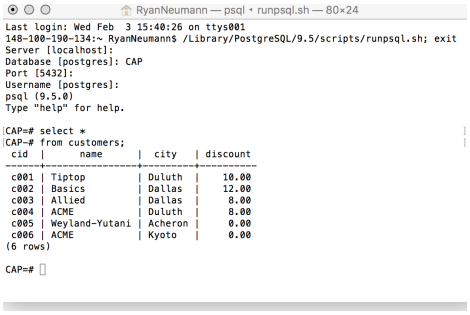
Ryan Neumann
Database Management
Pablo Rivas
4 February 2016



```
RyanNeumann — psql • runpsql.sh — 80×24
Last login: Wed Feb  3 15:40:26 on ttys001
148-100-190-134:~ RyanNeumann$ /Library/PostgreSQL/9.5/scripts/runpsql.sh; exit
Server [localhost]:
Database [postgres]: CAP
Port [5432]:
Username [postgres]:
psql (9.5.0)
Type "help" for help.

CAP=# select *
CAP-# from customers;
 cid  |      name       |  city   | discount
------+-----------------+---------+----------
 c001 | Tiptop          | Duluth  |    10.00
 c002 | Basics          | Dallas  |    12.00
 c003 | Allied          | Dallas  |     8.00
 c004 | ACME            | Duluth  |     8.00
 c005 | Weyland-Yutani  | Acheron |     0.00
 c006 | ACME            | Kyoto   |     0.00
(6 rows)

CAP=#
```

```
CAP=# select *
CAP-# from agents;
 aid | name  |   city   | percent
-----+-------+----------+---------
 a01 | Smith | New York |       6
 a02 | Jones | Newark   |       6
 a03 | Brown | Tokyo    |       7
 a04 | Gray  | New York |       6
 a05 | Otasi | Duluth   |       5
 a06 | Smith | Dallas   |       5
 a08 | Bond  | London   |       7
(7 rows)

CAP-#
```

```
(8 rows)

CAP=# select *
CAP-# from orders;
 ordno | mon | cid  | aid | pid | qty  | dollars
-------+-----+------+-----+-----+------+---------
  1011 | jan | c001 | a01 | p01 | 1000 |  450.00
  1013 | jan | c002 | a03 | p03 | 1000 |  880.00
  1015 | jan | c003 | a03 | p05 | 1200 | 1104.00
  1016 | jan | c006 | a01 | p01 | 1000 |  500.00
  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 |  740.00
(14 rows)
```

```
CAP=# select *
CAP-# from products;
 pid |  name  |  city   | quantity | priceusd
-----+--------+---------+----------+----------
 p01 | comb   | Dallas  |   111400 |     0.50
 p02 | brush  | Newark  |   203000 |     0.50
 p03 | razor  | Duluth  |   150600 |     1.00
 p04 | pen    | Duluth  |   125300 |     1.00
 p05 | pencil | Dallas  |   221400 |     1.00
 p06 | folder | Dallas  |   123100 |     2.00
 p07 | case   | Newark  |   100500 |     1.00
 p08 | clip   | Newark  |   200600 |     1.25
(8 rows)
```

A primary key has the smallest amount of super keys. Ideally only composed of one. A candidate key is a super key for which no proper subset is a super key. Lastly, a super key is simply a subset of a key.

A data type specifies the type of data of any object. All tables include data types. Such as a database table for the sales at a restaurant. There would be a field for the name of the food or beverage sold. This would be a text field and would not be nullable. Then the next field would likely be how many of the item was sold. This would be an int field and would be nullable. Then would come the price of the item. This would be a money data type, and would be nullable. After that, there would be a total sold, giving the total dollar amount of each item that was sold. This would be a money data type, and would be nullable.

The first normal form rule sets the basic rules for an organized database. It ensures that there are no repeating groups of data and also that there is a primary key. However, you must define data items. So you would put all the columns relating payment of businesses in the Business Payment table, and those relating to individual customers into the Customers table.

The second rule of the relational model, or the “access rows by content only” rule, can only retrieve rows by their content. There is no order on the row. Thus, a query cannot ask for an entire row of a table. Instead, you would have to specify the value of the row.

The third rule, “all rows must be unique”, stating that two rows may not have the same values in the entire column. This is to distinguish each row from one another, allowing a query language to retrieve it with it’s unique value. For example, in a Student table, the Student_Number values would have to be unique.