Tuesday 8 October 2013

General Assembly

Practical SQL

Operations

We can do simple arithmetic math

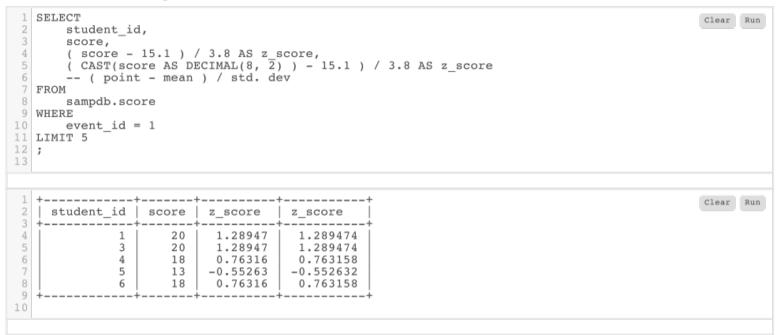
```
SELECT
                                                                                              Clear
                                                                                                   Run
       score / 100 as pct score,
       CAST(score AS decimal(8, 2)) / 100 as pct score 2
 4 FROM
       sampdb.score
6 LIMIT 5;
                                                                                              Clear Run
                 pct_score_2
     pct score
                  0.200000
       0.2000
       0.2000
                0.200000
0.180000
                  0.200000
       0.1800
       0.1300
                  0.130000
                    0.180000
       0.1800
10
```

⁻⁻ Note that MySQL will automatically convert the value returned as a decimal.

⁻⁻ Other SQL versions will not. Postgres requires a type conversion of the INT to a DECIMAL type.

More complex math

We can do more complex math



We are 'hard coding' the mean and standard deviation values

We'll discuss how to get these on the fly

Strings

String functions manipulate the appearance of text

```
SELECT
                                                                                                 Clear Run
       first name,
       UPPER(first name),
       LOWER (first name)
 6 FROM
       sampdb.president
 9 LIMIT 5;
10
                                                                                                 Clear
     first name
                  UPPER(first name)
                                       LOWER(first name)
     George
                  GEORGE
                                        george
     John
                                        john
                  JOHN
     Thomas
                  THOMAS
                                        thomas
     James
                  JAMES
                                        james
     James
                  JAMES
                                        james
10
```

String extraction

String functions can "reach into" strings

```
SELECT
                                                                                                Clear
                                                                                                     Run
       last name,
       LENGTH(last name) AS len,
       LEFT(last name, 3) AS lft,
       RIGHT(last name, 3) AS rght,
       SUBSTRING(last name, 3, 2) AS sub,
       SUBSTRING(last name, LENGTH(last name) - 2, 2) AS penultimate 2
 8 FROM
       sampdb.president
10 LIMIT 5;
11
                                                                                                Clear Run
     last name
                        lft
                               rght
                                            penultimate 2
                  len
                                      sub
     Washington
                   10
                        Was
                               ton
                                            to
     Adams
                        Ada
                               ams
                                      am
                                            am
     Jefferson
                                      ff
                        Jef
                               son
                                            SO
     Madison
                        Mad
                               son
                                      di
                                            so
     Monroe
                        Mon
                               roe
                                      nr
                                            ro
10
```

Finding strings allows more interesting usages

String manipulation becomes more interesting when we start looking for strings

```
SELECT
                                                                                               Clear
                                                                                                     Run
       email,
       -- LOCATE('@', email),
      position('@' in email)
  FROM
       sampdb.member
 7 LIMIT 5;
                                                                                               Clear
                                 position('@' in email)
     jeanne s@earth.com
     august.lundsten@pluto.com
                                                      16
     NULL
                                                    NULL
    arbogast.ruth@mars.net
                                                      14
     c.dorfman@uranus.net
                                                       10
10
```

Here we get the position of the @ sign

We can use that, with our other functions, to get a list of domains

Putting string functions together

With the location of the '@', we can extract the domain name



Now, for each record, we can extract the domain name

Might use this to count members by domain our list use, or check for correlation with other factors

Date Functions

```
SELECT
                                                                                                Clear Run
 2
       birth.
       death,
       YEAR(birth) AS yr,
       MONTH(birth) AS mnt,
       MONTHNAME (birth) AS mnt name,
       DATEDIFF (death, birth) AS days,
       DATEDIFF(death, birth) / 365 AS years
 9 FROM
       sampdb.president
10
11 LIMIT 5;
12
                                                                                                Clear Run
                  death
     birth
                                yr
                                       mnt
                                               mnt name
                                                          days
                                                                   years
     1732-02-22
                  1799-12-14
                                1732
                                               February
                                                          24767
                                                                   67.8548
     1735-10-30
                  1826-07-04
                                1735
                                               October
                                                                   90.7370
                                          10
                                                          33119
                                                                   83.2795
     1743-04-13
                  1826-07-04
                                1743
                                               April
                                                          30397
     1751-03-16
                                1751
                                                                   85.3425
                  1836-06-28
                                               March
                                                          31150
                  1831-07-04
                                1758
     1758-04-28
                                               April
                                                          26729
                                                                   73.2301
10
```

Putting it together

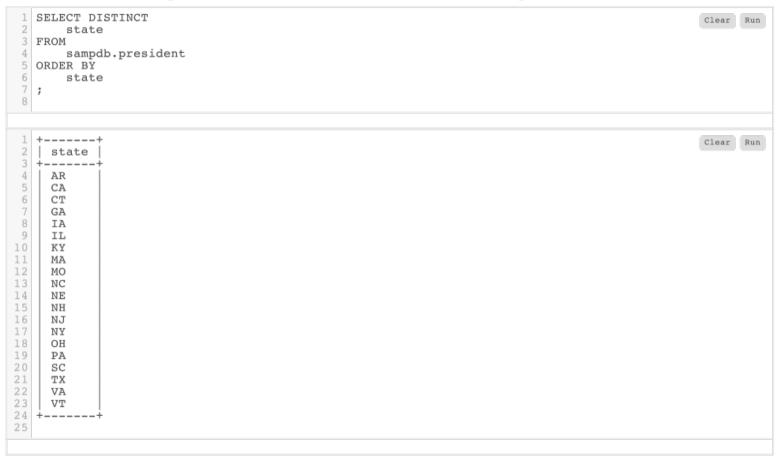
We can combine records, operations and functions to calculate and format output

```
SELECT
                                                                                                Clear Run
       CONCAT (
           first name,
           last name,
           " lived ",
           ROUND(DATEDIFF(death, birth) / 365, 1),
           " vears"
       ) AS sentence
10 FROM
       sampdb.president
12 LIMIT 5;
13
                                                                                               Clear Run
     sentence
     George Washington lived 67.9 years
     John Adams lived 90.7 years
    Thomas Jefferson lived 83.3 years
    James Madison lived 85.3 years
    James Monroe lived 73.2 years
10
```

More on Aggregate Functions

We attack this through dataset exploration

SELECT DISTINCT gives us the set of values in an attribute -- no repetitions



This is basically what we get from GROUP BY

GROUP BY subdivides the dataset into subsets

Each subset is characterized by having the same value for the GROUP BY attribute

```
1 SELECT
                                                                                               Clear Run
       state,
       COUNT(state) AS count
 4 FROM
       sampdb.president
 6 GROUP BY
       state
 8 ORDER BY
 9
       state
10;
11
                                                                                               Clear Run
     state
             count
     AR
     CA
 6
     GA
     IΑ
     IL
     KY
     MA
                 4
12
     MO
13
     NC
14
     NE
15
     NH
16
     NJ
     NY
18
     OH
19
     PA
20
     SC
21
     TX
22
     VA
                 8
23
     VT
24 +----
25
```

Same analysis, against member

```
SELECT
                                                                                                      Clear
                                                                                                           Run
       state,
       COUNT(state) AS count
 4 FROM
       sampdb.member
 6 GROUP BY
       state
 8 ORDER BY
       state
10 LIMIT 15;
11
                                                                                                      Clear Run
     state
              count
     AK
     AL
                   3
     ΑZ
     CA
                   6
     CO
10
     FL
11
     GA
12
     HΙ
13
     ΙA
14
     ID
15
     IL
                   6
16
     IN
                   3
17
     KS
18
     ΚY
20
```

DISTINCT also works by _tuples_

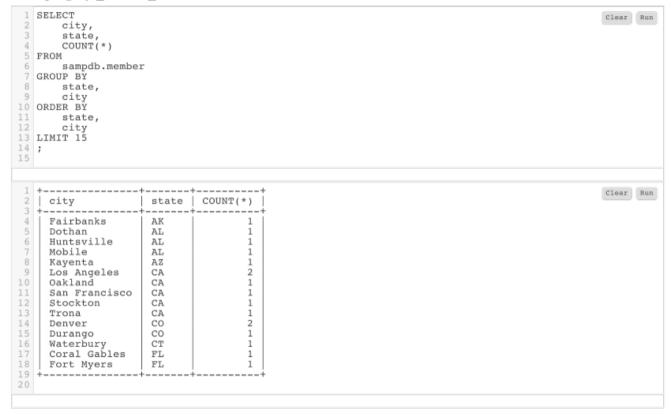
```
SELECT DISTINCT
                                                                                                   Clear
                                                                                                         Run
       city,
       state
  FROM
       sampdb.member
  ORDER BY
       state,
       city
 9 LIMIT 15
10 ;
11
                                                                                                   Clear Run
     city
                      state
     Fairbanks
                      AK
     Dothan
                      AL
     Huntsville
                      AL
     Mobile
                      AL
     Kayenta
                      AZ
     Los Angeles
                      CA
10
     Oakland
                      CA
11
     San Francisco
12
     Stockton
                      CA
13
     Trona
                      CA
14
     Denver
                      CO
15
     Durango
                      CO
16
     Waterbury
17
     Coral Gables
                      FL
18
     Fort Myers
19
20
```

Here we are getting a list of city / state value pairs that occur at least once in the data

This is not _permutations_, e.g. the possible combinations of these values -- each of these _does_ appear at least once

As does GROUP BY

Grouping by _several_ attributes subdivides counts



All the Alabama records are still in the output.

But the sum of Alabama records, which was 3, is now subdivided over three different cities

ORDER BY shows the grouping of the records

ELECT					Clea
last_name					
first_nam	me,				
member_i	d,				
city,					
state					
ROM					
sampdb.me RDER BY	ember				
state,					
city					
MIT 15					
THILL ID					
	+	+	·	++	Clea
last_name	first_name	member_id	city	state	
Matthews	Bill	56	Fairbanks	AK	
Edwards	John	82	Dothan	AL	
Hughes	Max	37	Huntsville	AL	
Schauer	Alma	65	Mobile	AL	
Kirby	Timothy	48	Kayenta	AZ	
Puntillo	Cheryl	59	Los Angeles	CA	
		88	Los Angeles	CA	
Pierson	Stanley				
Pierson Sprague	Earl	99	Oakland	CA	
Pierson Sprague Smith	Earl Laura	99 98	Oakland San Francisco	CA CA	
Pierson Sprague Smith Feit	Earl Laura Daniel	99 98 19	Oakland San Francisco Stockton	CA CA CA	
Pierson Sprague Smith Feit Simmons	Earl Laura Daniel David	99 98 19 49	Oakland San Francisco Stockton Trona	CA CA CA	
Pierson Sprague Smith Feit Simmons Garner	Earl Laura Daniel David Steve	99 98 19 49 89	Oakland San Francisco Stockton Trona Denver	CA CA CA CA	
Pierson Sprague Smith Feit Simmons Garner Sawyer	Earl Laura Daniel David Steve Dennis	99 98 19 49 89 7	Oakland San Francisco Stockton Trona Denver Denver	CA CA CA CA CO	
Pierson Sprague Smith Feit Simmons Garner Sawyer Bookstaff	Earl Laura Daniel David Steve Dennis Barbara	99 98 19 49 89 7 47	Oakland San Francisco Stockton Trona Denver Denver Durango	CA CA CA CA CO CO	
Pierson Sprague Smith Feit Simmons Garner Sawyer	Earl Laura Daniel David Steve Dennis	99 98 19 49 89 7	Oakland San Francisco Stockton Trona Denver Denver	CA CA CA CA CO	
Pierson Sprague Smith Feit Simmons Garner Sawyer Bookstaff	Earl Laura Daniel David Steve Dennis Barbara	99 98 19 49 89 7 47	Oakland San Francisco Stockton Trona Denver Denver Durango	CA CA CA CA CO CO	

Referring to SQL results in SQL queries -- Subqueries

Subqueries allow us to embed one query within another

These embedded queries are 'subqueries'

We have to watch the table returned by the subquery carefully

The returned table has to have the fields and row expected of it by the calling query

Two varieties, 'correlated' and 'uncorrelated'

'correlated' is cooler but harder, we'll do 'uncorrelated' first

Presidents from the most presidential states

Say we want a list of those presidents from the three states that have sent the most presidents

We can get the list by a query:



Hard-code the values

We can put those values into a query

```
1 SELECT
                                                                                                    Clear Run
       last name,
       first name,
       state
  5 FROM
        sampdb.president
 7 WHERE
       state IN
             'VA',
            'ОН',
            'MA'
14 ORDER BY
15 state
16 ;
17
                                                                                                    Clear Run
 2 | last name
                   first name
     Bush
     Adams
                   John Quincy
                                     MA
     Kennedy
                   John F.
                                     MA
                                     MA
     Adams
                   John
                                     OH
     Harding
                   Warren G.
                                     OH
     Taft
                   William H.
     McKinley
                                     OH
                   William
                   Benjamin
                                     OH
     Harrison
                                     OH
     Garfield
                   James A.
                   Rutherford B.
                                     OH
     Hayes
14
     Grant
                   Ulysses S.
                                     OH
     Taylor
Tyler
                                     VA
                   Zachary
                   John
                                     VA
     Harrison
                   William H.
                                     VA
18
     Monroe
                   James
                                     VA
     Wilson
                   Woodrow
                                     VA
20 Mi
21 Je
22 Wi
23 +---
                                     VA
     Madison
                   James
                                     VA
     Jefferson
                   Thomas
     Washington
                   George
```

But, Kludgy!

What if the data changes?

1	Clear Run
1	Clear Run

1	Clear	Run
1	Clear	Run

1	Clear	Run
1	Clear	Run

Contact

gordon@practicalhorseshoeing.com