Chapter 5

October 29, 2019

```
In [1]: %load_ext sql

In [2]: %sql postgresql://postgres:postgres@localhost:5432/analysis

Out[2]: 'Connected: postgres@analysis'
```

1 Basic Math and Stats with SQL

- two integers return an integer
- a numeric on either side of the operator returns a numeric
- anything with floating point number returns a floating-point of type double precision

```
In [9]: %%sql

        SELECT 2 + 2;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[9]: [(4,)]

In [8]: %%sql

        SELECT 9 - 1;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[8]: [(8,)]

In [17]: %%sql

         SELECT 3 * 4;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.
```

```
Out[17]: [(12,)]
```

- we have done basic math operations

```
In [20]: %%sql

         SELECT 11 / 6;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[20]: [(1,)]
```

- division of an integer with another integer yields to an integer
- this here is basically 1 with a reminder of 5

```
In [21]: %%sql

         SELECT 11 % 6;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[21]: [(5,)]
```

- as seen here we get the remainder of the operation which we can do with the modulo operator
- if you want to check whether a number is evem, you can test it using the % 2 operation, if the result is 0 with no remainder, the number is even

```
In [22]: %%sql

         SELECT 11.0 / 6;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[22]: [(Decimal('1.8333333333333333'),)]
```

- the first one is a numeric
- so we get a numeric result

```
In [23]: %%sql

         SELECT CAST (11 AS numeric(3,1)) / 6;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.
```

```
Out[23]: [(Decimal('1.8333333333333333'),)]
```

- we can force the 11 with the CAST function into a numeric

```
In [26]: %%sql

         SELECT 3 ^ 4;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[26]: [(81.0,)]
```

- exponent

```
In [27]: %%sql

         SELECT sqrt(10);

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[27]: [(3.1622776601683795,)]
```

- sqrt(n) function

1.1 Doing Math Across Columns

```
In [35]: %%sql

         SELECT geo_name,
                state_us_abbreviation AS "st",
                p0010001 AS "Total Population",
                p0010003 AS "White Alone",
                p0010004 AS "Black or African American Alone",
                p0010005 AS "Am Indian/Alaska Native Alone",
                p0010006 AS "Asian Alone",
                p0010007 AS "Native Hawaiian and Other Pacific Islander Alone",
                p0010008 AS "Some Other Race Alone",
                p0010009 AS "Two or More Races"
         FROM us_counties_2010
         LIMIT 5;

 * postgresql://postgres:***@localhost:5432/analysis
5 rows affected.
```

```
Out[35]: [('Autauga County', 'AL', 54571, 42855, 9643, 232, 474, 32, 466, 869),
          ('Baldwin County', 'AL', 182265, 156153, 17105, 1216, 1348, 89, 3631, 2723),
          ('Barbour County', 'AL', 27457, 13180, 12875, 114, 107, 29, 894, 258),
          ('Bibb County', 'AL', 22915, 17381, 5047, 64, 22, 13, 185, 203),
          ('Blount County', 'AL', 57322, 53068, 761, 307, 117, 38, 2347, 684)]
```

- we select the table
- this table we give each column name (the p name) an alias
- this alias we give with the *AS* keyword which will make the list more readable
- we limit the list to 5 because it has more than 3000 rows!!!

```
In [36]: %%sql

         SELECT geo_name,
                state_us_abbreviation AS "st",

                p0010003 AS "White Alone",
                p0010004 AS "Black Alone",
                 p0010003 + p0010004 AS "Total While and Black"
         FROM us_counties_2010
         LIMIT 5;
```

```
 * postgresql://postgres:***@localhost:5432/analysis
5 rows affected.
```

```
Out[36]: [('Autauga County', 'AL', 42855, 9643, 52498),
          ('Baldwin County', 'AL', 156153, 17105, 173258),
          ('Barbour County', 'AL', 13180, 12875, 26055),
          ('Bibb County', 'AL', 17381, 5047, 22428),
          ('Blount County', 'AL', 53068, 761, 53829)]
```

- **Adding and Subtracting Columns**
- we simply add two rece columns together
- we identified both
- we are doing a basic add operation with the both column names
- we use again an *AS* keyword for the result of this new line, which is our result

```
In [45]: %%sql

         SELECT geo_name,
                state_us_abbreviation AS "st",
                p0010001 AS "Total",
                 p0010003 + p0010004 + p0010005 + p0010006 + p0010007
                 + p0010008 + p0010009 AS "All Races",
                 (p0010003 + p0010004 + p0010005 + p0010006 + p0010007
                 + p0010008 + p0010009) - p0010001 AS "Difference"

         FROM us_counties_2010
```

```
        ORDER BY "Difference"
        LIMIT 5;

 * postgresql://postgres:***@localhost:5432/analysis
5 rows affected.


Out[45]: [('Baldwin County', 'AL', 182265, 182265, 0),
          ('Barbour County', 'AL', 27457, 27457, 0),
          ('Bibb County', 'AL', 22915, 22915, 0),
          ('Blount County', 'AL', 57322, 57322, 0),
          ('Autauga County', 'AL', 54571, 54571, 0)]
```

- we first have the total population as Total
- we add all 7 races which we define as All Races
- the total population and the calculated one (All races) are equal
- the difference should be 0
- we order by difference
- whenever we import some data we can do this test to be sure that there is no calculation bias

```
In [51]: %%sql

        SELECT geo_name, state_us_abbreviation AS "st",
        (CAST (p0010006 AS numeric(8,1)) / p0010001) * 100 AS "pct_asian"
        FROM us_counties_2010
        ORDER BY "pct_asian" DESC
        LIMIT 5;

 * postgresql://postgres:***@localhost:5432/analysis
5 rows affected.


Out[51]: [('Honolulu County', 'HI', Decimal('43.89497769109962474000')),
          ('Aleutians East Borough', 'AK', Decimal('35.97580388411333970100')),
          ('San Francisco County', 'CA', Decimal('33.27165361664607226500')),
          ('Santa Clara County', 'CA', Decimal('32.02237037519322063600')),
          ('Kauai County', 'HI', Decimal('31.32461880132953749400'))]
```

- we devide the asian value which we cast to a numeric and the total population

- **Tracking Percent Change**

```
In [53]: %%sql

        CREATE TABLE percent_change (
        department varchar(20),
        spend_2014 numeric(10,2),
        spend_2017 numeric(10,2)
        );
```

```
 * postgresql://postgres:***@localhost:5432/analysis
Done.


Out[53]: []

In [55]: %%sql

         INSERT INTO percent_change
         VALUES
             ('Building', 250000, 289000),
             ('Assessor', 178556, 179500),
             ('Library', 87777, 90001),
             ('Clerk', 451980, 650000),
             ('Police', 250000, 223000),
             ('Recreation', 199000, 195000);

 * postgresql://postgres:***@localhost:5432/analysis
6 rows affected.


Out[55]: []

In [59]: %%sql

         SELECT department,
         spend_2014,
         spend_2017,
         round( (spend_2017 -  spend_2014) / spend_2014 * 100, 1) AS "pct_change"
         FROM percent_change
         ORDER BY "pct_change" DESC;

 * postgresql://postgres:***@localhost:5432/analysis
6 rows affected.


Out[59]: [('Clerk', Decimal('451980.00'), Decimal('650000.00'), Decimal('43.8')),
          ('Building', Decimal('250000.00'), Decimal('289000.00'), Decimal('15.6')),
          ('Library', Decimal('87777.00'), Decimal('90001.00'), Decimal('2.5')),
          ('Assessor', Decimal('178556.00'), Decimal('179500.00'), Decimal('0.5')),
          ('Recreation', Decimal('199000.00'), Decimal('195000.00'), Decimal('-2.0')),
          ('Police', Decimal('250000.00'), Decimal('223000.00'), Decimal('-10.8'))]
```

- we make a SELECT statement with the required columns
- we use the round function which has two arguments
  - the value (which is our division operation) and the precision of 1, which is the value after the , (here we are doing a rounding)
- we give our operation an Alias pct_change

- here our operation was to see the percentual difference between two values
- we can now see that the clerk department has spend nearly 43 percent more in 2017 than in 2014

- **Aggregate Functions for Average and Sums**

```
In [60]: %%sql

         SELECT sum(p0010001) AS "County Sum",
         round(avg(p0010001), 0) AS "County Average"
         FROM us_counties_2010;
```

```
 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.
```

```
Out[60]: [(308745538, Decimal('98233'))]
```

- this one will compare the sum of the countys population
- with the average of the countys population
- we do this with the sum() and avg() functions

1.2 Finding the Median

- **average** the sum of all the values divided by the number of values
- **median** the middle value of an ordered set of values
- a good test is to take both
  - if they are close the distributiion is a normal one

```
In [63]: %%sql

         DROP TABLE percentile_test;
```

```
 * postgresql://postgres:***@localhost:5432/analysis
Done.
```

```
Out[63]: []
```

```
In [64]: %%sql

         CREATE TABLE percentile_test (
         numbers integer
         );
```

```
 * postgresql://postgres:***@localhost:5432/analysis
Done.
```

```
Out[64]: []
```

```
In [66]: %%sql

         INSERT INTO percentile_test (numbers) VALUES
         (1), (2), (3), (4), (5), (6);

 * postgresql://postgres:***@localhost:5432/analysis
6 rows affected.


Out[66]: []
```

```
In [69]: %%sql

         SELECT
         percentile_cont(.5)
         WITHIN GROUP (ORDER BY numbers),
         percentile_disc(.5)
         WITHIN GROUP (ORDER BY numbers)
         FROM percentile_test;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[69]: [(3.5, 3)]
```

- there is no mediam() function in sql
- we can however build it with the percentile functions which are two
- we have two percentile functions _cont() and _disc()
- the median is equivalent to the 50th percentile, half the values are below and half above
- the _cont() function calculates as continuous values
  - the median is bewteen 3 and 4 and is 3.5
- the _disc() function calculates as discrete values
  - the value will be rounded to one of the 2 values

```
In [73]: %%sql

         SELECT sum(p0010001) AS "County Sum",
         round(avg(p0010001), 0) AS "County Average",
         percentile_cont(.5)
         WITHIN GROUP (ORDER BY p0010001) AS "County Median"
         FROM us_counties_2010

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[73]: [(308745538, Decimal('98233'), 25857.0)]
```

- **finding other quantiles with percentile functions**

```
In [77]: %%sql

         SELECT percentile_cont(array[.25, .5, .75])
         WITHIN GROUP (ORDER BY p0010001) AS "quartiles"
         FROM us_counties_2010;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[77]: [([11104.5, 25857.0, 66699.0],)]
```

```
In [78]: %%sql

         SELECT unnest(
         percentile_cont(array[.25, .5, .75])
         WITHIN GROUP (ORDER BY p0010001)
         ) AS "quartiles"
         FROM us_counties_2010;

 * postgresql://postgres:***@localhost:5432/analysis
3 rows affected.


Out[78]: [(11104.5,), (25857.0,), (66699.0,)]
```

- we created an array of quartiles cut points
- those points we pass into our percentile_cont() function
- we use here unnest function to make the result output in each rows
- this means 25% of the counties has as population of 11,104.5 or less

```
In [95]: %%sql

         CREATE OR REPLACE FUNCTION _final_median(anyarray)
            RETURNS float8 AS
         $$
           WITH q AS
           (
              SELECT val
              FROM unnest($1) val
              WHERE VAL IS NOT NULL
              ORDER BY 1
           ),
           cnt AS
           (
             SELECT COUNT(*) AS c FROM q
           )
```

```
          SELECT AVG(val)::float8
          FROM
          (
            SELECT val FROM q
            LIMIT  2 - MOD((SELECT c FROM cnt), 2)
            OFFSET GREATEST(CEIL((SELECT c FROM cnt) / 2.0) - 1,0)
          ) q2;
        $$
        LANGUAGE sql IMMUTABLE;
```

```
 * postgresql://postgres:***@localhost:5432/analysis
Done.
```

```
Out[95]: []
```

```
In [94]: %%sql

        CREATE AGGREGATE median(anyelement) (
          SFUNC=array_append,
          STYPE=anyarray,
          FINALFUNC=_final_median,
          INITCOND='{}'
        );
```

```
 * postgresql://postgres:***@localhost:5432/analysis
Done.
```

```
Out[94]: []
```

- we create an median() function
- note that functions will be covered later
- just take this function as given

```
In [99]: %%sql

        SELECT sum(p0010001) AS "County Sum",
               round(AVG(p0010001), 0) AS "County Average",
               median(p0010001) AS "County Median",
               percentile_cont(.5)
               WITHIN GROUP (ORDER BY P0010001) AS "50th Percentile"
        FROM us_counties_2010;
```

```
 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.
```

```
Out[99]: [(308745538, Decimal('98233'), 25857.0, 25857.0)]
```

- we simply add the median function
- and we compare it with the pecentile_cont function which has .5 value
- both are equal

1.3 Finding the Mode

```
In [101]: %%sql

          SELECT mode() WITHIN GROUP (ORDER BY p0010001)
          FROM us_counties_2010;

 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.


Out[101]: [(21720,)]
```

1.4 Tasks

– 2. Using the 2010 Census county data, find out which New York state county – has the highest percentage of the population that identified as “American – Indian/Alaska Native Alone.” What can you learn about that county from online – research that explains the relatively large proportion of American Indian – population compared with other New York counties?

```
In [113]: %%sql

          SELECT
              geo_name,
              state_us_abbreviation,
              p0010001 AS "Total Population",
              p0010005 AS "Am Indian/Alaska Native Alone",
              (CAST (p0010005 AS numeric(8,1)) / p0010001) * 100
              AS percent_american_indian_alaska_native_alone
          FROM
              us_counties_2010
          WHERE
              state_us_abbreviation  = 'NY'
          ORDER BY
              percent_american_indian_alaska_native_alone DESC
          LIMIT
              5;

 * postgresql://postgres:***@localhost:5432/analysis
5 rows affected.


Out[113]: [('Franklin County', 'NY', 51599, 3797, Decimal('7.35866974166166011000')),
           ('Cattaraugus County', 'NY', 80317, 2443, Decimal('3.04169727454959722100')),
           ('Bronx County', 'NY', 1385108, 18260, Decimal('1.31830875281927474200')),
           ('Genesee County', 'NY', 60079, 679, Decimal('1.13017859817906423200')),
           ('Niagara County', 'NY', 216469, 2285, Decimal('1.05557839690671643500'))]
```

– 3. Was the 2010 median county population higher in California or New York?

```
In [114]: %%sql

          SELECT
              percentile_cont(.5)
              WITHIN GROUP (ORDER BY p0010001)
          FROM
              us_counties_2010
          WHERE
              state_us_abbreviation = 'NY';
```

```
 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.
```

```
Out[114]: [(91301.0,)]
```

```
In [115]: %%sql

          SELECT
              percentile_cont(.5)
              WITHIN GROUP (ORDER BY p0010001)
          FROM
              us_counties_2010
          WHERE
              state_us_abbreviation = 'CA';
```

```
 * postgresql://postgres:***@localhost:5432/analysis
1 rows affected.
```

```
Out[115]: [(179140.5,)]
```

```
In [118]: %%sql

          SELECT
              state_us_abbreviation,
              percentile_cont(0.5)
              WITHIN GROUP (ORDER BY p0010001) AS median
          FROM
              us_counties_2010
              GROUP BY state_us_abbreviation
          LIMIT
              5;
```

```
 * postgresql://postgres:***@localhost:5432/analysis
5 rows affected.
```

```
Out[118]: [('AK', 7029.0),
           ('AL', 34339.0),
           ('AR', 19019.0),
           ('AZ', 131346.0),
           ('CA', 179140.5)]
```

- **median for each state**