Tutorial of week 11 - Function

Designed by ZHU Yueming, referenced by the teaching materials of Stephane Faroult.

Experimental-Objective

1. Introduce more functions in postgres
2. Learn how to create your function

Other Postgres Defined Functions:

1. generate_series()

```
SELECT * FROM generate_series(1,30);
SELECT * FROM generate_series(5,1,-2);
SELECT * FROM generate_series(4,3);
SELECT * FROM generate_series(4,5);
```

2. length()

- It will return the length of a string.

3. splite_part()

- Split string on delimiter and return the given field (counting from one)

  ```
  splite_part(varchar <source text>, varchar <delimiter text>,int <field
  serial number>)
  ```

- **Exercise 1:** If you need to split the title `Feel relaxed studying database` by a space into 4 different rows, what you plan to do?
  - Try following queries:

  ```
  select split_part('Feel relaxed studying database',' ',1);
  select split_part('Feel relaxed studying database',' ',2);
  select split_part('Feel relaxed studying database',' ',3);
  select split_part('Feel relaxed studying database',' ',4);
  ```

  It will returns only one row in seperate result set.

- Suppose `select split_part('Feel relaxed studying database',' ',n);` as a table named t1, and `generate_series(1, 4);` as a table named t2, what the result set of cross join of those two table?

  Try following queries:

```
select split_part(t1.words, ' ', t2)
from (select cast('Feel relaxed studying database' as text) as words) t1
         cross join generate_series(1, 4) t2;
```

We can replace 4 with `length(t1.words)-length(replace(t1.words,' ',''))+1`

```
select split_part(t1.words, ' ', t2)
from (select cast('Feel relaxed studying database' as text) as words) t1
         cross join generate_series(1, length(t1.words)-
length(replace(t1.words,' ',''))+1) t2;
```

4. substr()

- Get the substring

```
substr(varchar <source text>, int <begin position>, int <length>)
```

- **Exercise 2**: Design a query to separate all characters in `Feel relaxed studying database`

```
select distinct substr(t1.words,t2,1)
from (select cast('Feel relaxed studying database' as text) as words) t1
         cross join generate_series(1, length(t1.words)) t2;
```

5. ascii()

- Convert a character to its corresponding ASCII code

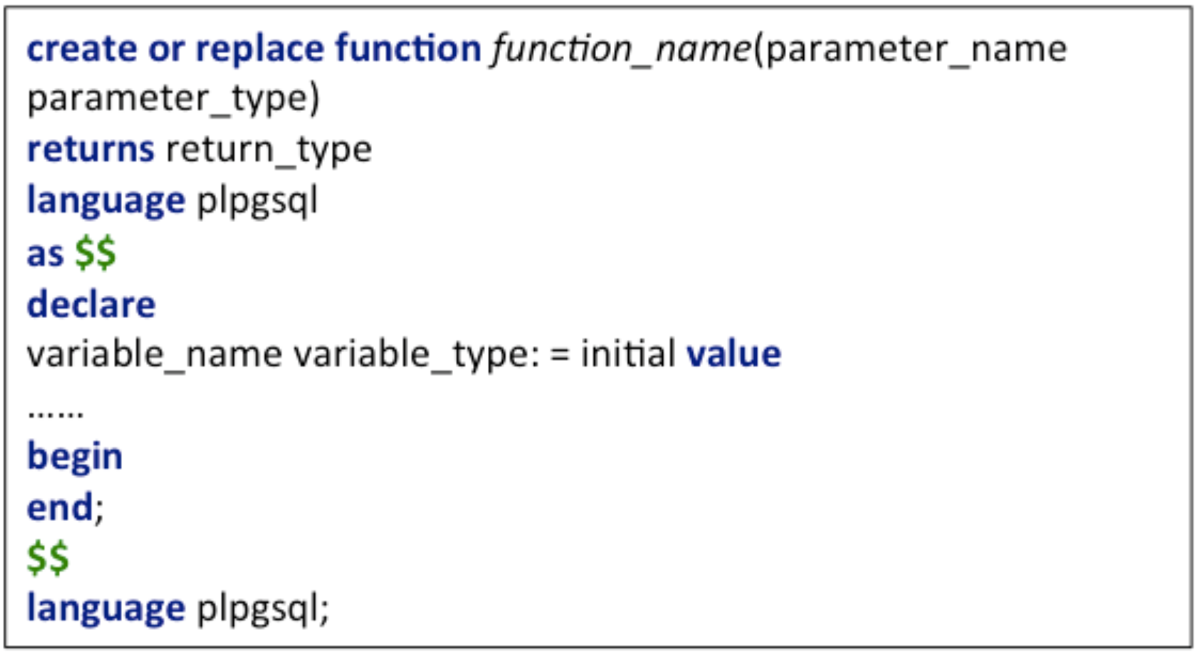
```
ascii(char <source char>)
```

- Exercise: Design a query to find all characters and their ascii code in `Feel relaxed studying database` in ascending order of ascii code.

```
select distinct ascii(substr(t1.words,t2,1)) as
ascii,substr(t1.words,t2,1)
from (select cast('Feel relaxed studying database' as text) as words) t1
         cross join generate_series(1, length(t1.words)) t2;
```

Function

1. general format of postgreSQL function



```
create or replace function function_name(parameter_name
parameter_type)
returns return_type
language plpgsql
as $$
declare
variable_name variable_type: = initial value
......
begin
end;
$$
language plpgsql;
```

Exercise 1: Create a function to calculate the sum of two integer numbers. After your design, you can execute the following query.

```
create or replace function sum_func(a int, b int)
  returns int
language plpgsql
as $function$
begin
  return a + b;
end;
$function$;
```

Test:

```
select fun(3,4);
```

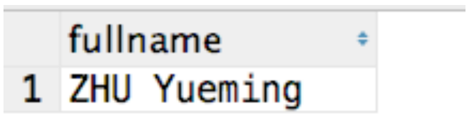
Exercise 2: Create a function named “fullname”, which has two variables called “firstname” and “secondname” and return the combination of two variables. After your design, you can execute following queries.

```
create or replace function fullname(firstname varchar, secondname varchar)
  returns varchar
language plpgsql
as $function$
declare
  name varchar :=null;
begin
  name := firstname || ' ' || secondname;
  return name;
end;
$function$;
```

Test:

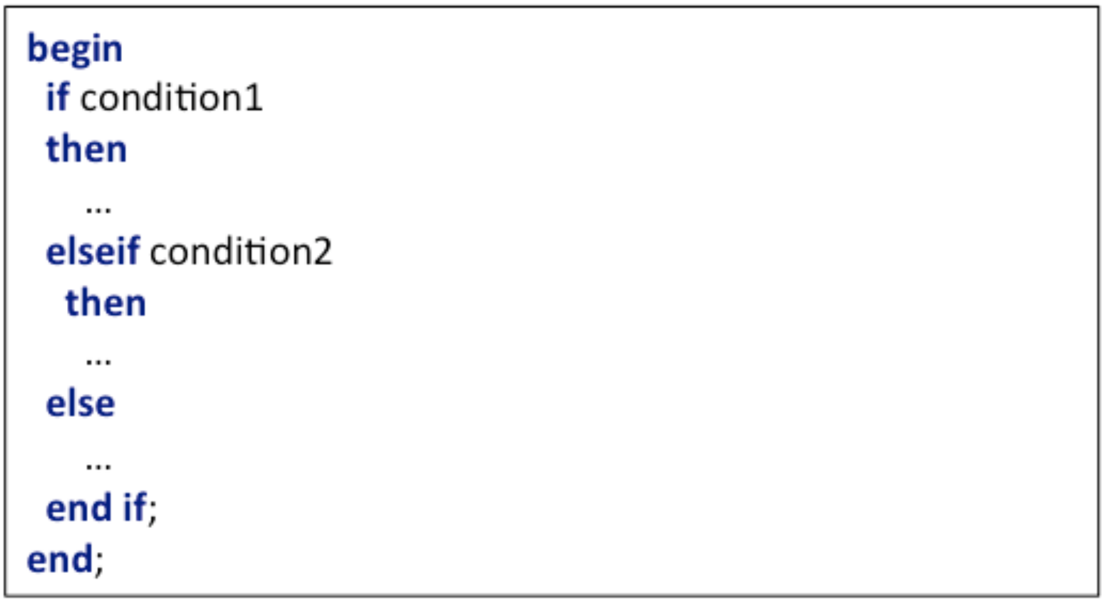
```
select fullname('ZHU','Yueming');
```

Result:



```
  fullname
1 ZHU Yueming
```

2. Conditions in Function



```
begin
  if condition1
  then
     …
  elseif condition2
   then
     …
  else
     …
  end if;
end;
```

Exercise 3: Create a function to combine firstname and surname of people according to the people coming from eastern country or western country.

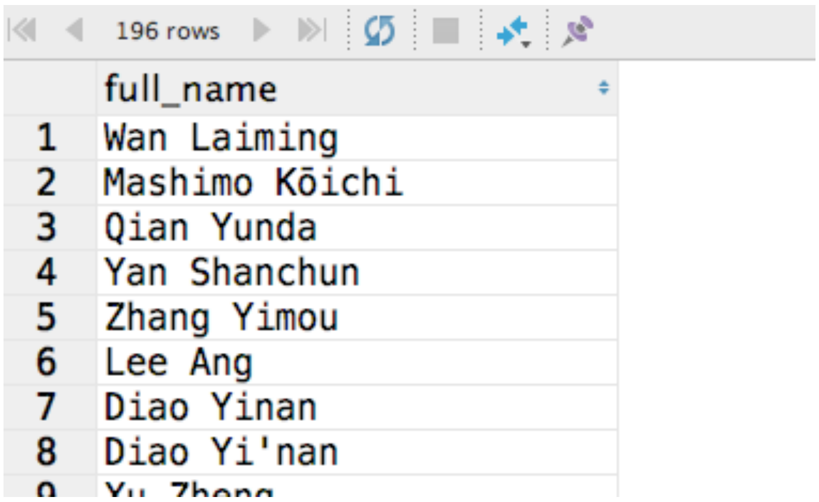
```
create function full_name(p_fn varchar, p_sn varchar, style char)
  returns varchar
as $$
begin
  if upper(style) = 'W'
  then
```

```
    return trim(coalesce(p_fn, '') || ' ' || p_sn);
  elseif upper(style) = 'E'
    then
      return trim(p_sn || ' ' || coalesce(p_fn, ''));
  else
    raise exception 'Style must be W or E';
  end if;
end;
$$
language plpgsql;
```

Test:

```
select full_name(p.first_name, p.surname, 'E')
from people p
  join credits c on p.peopleid = c.peopleid and c.credited_as = 'D'
  join movies m on m.movieid = c.movieid
where m.country = 'cn';
```

Result:



196 rows

	full_name
1	Wan Laiming
2	Mashimo Kōichi
3	Qian Yunda
4	Yan Shanchun
5	Zhang Yimou
6	Lee Ang
7	Diao Yinan
8	Diao Yi'nan
9	Yu Zheng

3. Loop in function

```
for variable_value in start_value .. end_value loop
  statements;
end loop;

while condition loop
  statements;
end loop;
```

Exercise 4: Find the factorial of number

```
create or replace function factorial(number int)
  returns int
language plpgsql
as $function$
declare result int;
begin
  result = 1;
  for i in 1 .. number loop
    result = result * i;
  end loop;
  return result;
end;
$function$;
```

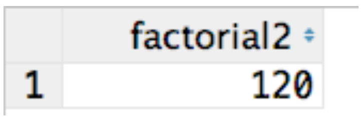
or

```
create or replace function factorial2(number int)
  returns int
language plpgsql
as $function$
declare
  result int;
  i      int;
begin
  result = 1;
  i = 1;
  while i <= number loop
    result = result * i;
    i = i + 1;
  end loop;
  return result;
end;
$function$;
```

Test:

```
select factorial2(5);
```

Result:



	factorial2
1	120

4. Return A table from Function

```
create function fun_name(arg1 type1, ......)
   returns
      table
      (
         col_name1 col_type,
         col_name2 col_type,
             ......
      )
as
$$
begin
   return query select col1, col2 from .....;
end;
$$
language plpgsql;
```

The column type of result set should be same as the type of return table exactly, more specifically, the type of col1 should be same as the first col_type, and the type of col2 should be same as the second col_type.

Exercise 5: Design a function to return a table that contains all characters and their ascii code from a pattern string in ascending order of ascii code.

```
create function character_table(pattern varchar)
    returns table
            (
                chr   char,
                ascii int
            )
as
$body$
begin
    return query
        select distinct cast(substr(t1.title, t2, 1) as char) chr,
ascii(substring(t1.title, t2, 1)) ascii
        from (select pattern as title) t1
                 cross join generate_series(1, length(pattern)) t2
        order by ascii;
end;
$body$
```

```
    LANGUAGE plpgsql;
```

Then you can test the function as

```
select * from character_table('I love database!');
```

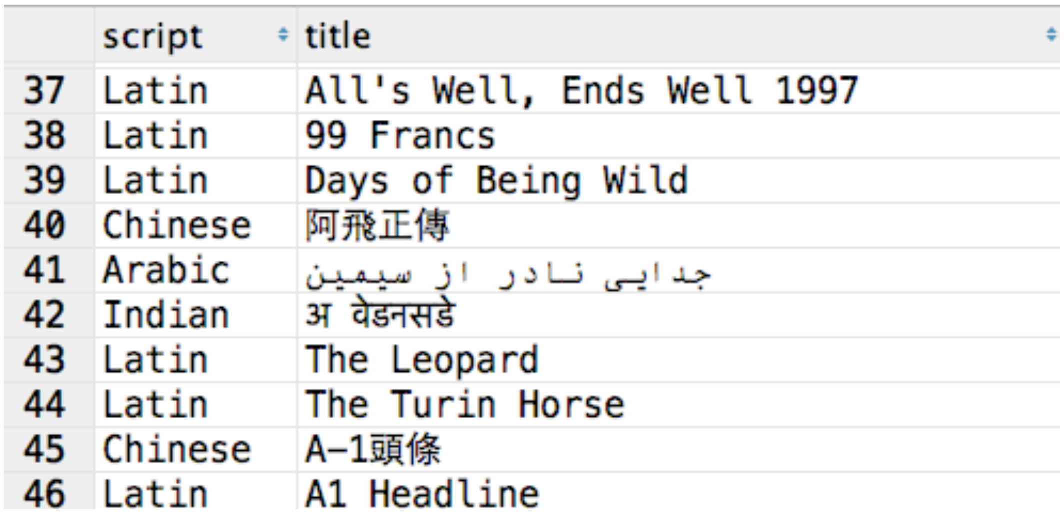
5. Comprehensive Example (Provide by Stephane Faroult)

Writing a function that recognizes in which script a text is written (to be applied to column TITLE in table ALT_TITLES). We'll only consider the main writing systems.
Here is a reference:
link

In particular the table at List of writing scripts by adoption, with the number of users.
If you execute the query:

```
select script (title ),title from (select title from alt_titles) x
```

The result would be



	script	title
37	Latin	All's Well, Ends Well 1997
38	Latin	99 Francs
39	Latin	Days of Being Wild
40	Chinese	阿飛正傳
41	Arabic	جدایی نادر از سیمین
42	Indian	अ वेडनसडे
43	Latin	The Leopard
44	Latin	The Turin Horse
45	Chinese	A-1頭條
46	Latin	A1 Headline

Hints:
Ranges to consider for the ascii() return value (approximate blocks but the result should be OK - can be refined if needed)
Latin
<=740
[7424,8594]
[11360,11391]
[42786,43876]

Greek
[880, 1023]
[7462, 8446]

Cyrillic
[1024, 1327]
[7296, 7544]
[42560, 42655]

Arabic
[1536, 2303]
[64336, 69246]
[124464, 126705]

Indian
[2304, 3572]

Thai
[3585, 3675]

Burmese
[4096, 4255]

Korean
[4352, 4607]
[12593, 12686]
[12800, 12926]
[43360, 55203]
[43360, 55291]
[65440, 65500]

Khmer
[6016, 6137]

Chinese
[11904, 12333]
[12344, 12347]
[13312, 42182]

Japanese
[12353, 12543]
[12784, 12799]
[13008, 13143]

Other
 everything else ...

Solution1:

```
create function script(fm character varying)
  returns character varying
language plpgsql
as $$
declare
  ascimax int;
  ascimin int;
  value   varchar;
begin
  select max(x.a)
  into ascimax
  from (
         select distinct
           ascii(substr(t.title, n, 1)) a,
           substr(t.title, n, 1)
         from (select fm as title) t
           cross join generate_series(1, length(t.title)) n) x;
  select min(x.a)
  into ascimin
  from (
         select distinct
           ascii(substr(t.title, n, 1)) a,
           substr(t.title, n, 1)
         from (select fm as title) t
           cross join generate_series(1, length(t.title)) n) x
  where a > 127;
  if ((ascimax <= 740 and ascimax >= 0) or (ascimax >= 7424 and ascimax <=
8594) or
      (ascimax >= 11360 and ascimax <= 11391) or (ascimax >= 42786 and ascimax
<= 43876))
  then value = 'Latin';
  elseif ((ascimin >= 880 and ascimin <= 1023) or (ascimin >= 7462 and ascimin
<= 8446))
    then value = 'Greek';
  elseif ((ascimin >= 1024 and ascimin <= 1327) or (ascimin >= 7296 and ascimin
<= 7544) or
          (ascimin >= 42560 and ascimin <= 42655))
    then value = 'Cyrillic';
  elseif ((ascimin >= 1536 and ascimin <= 2303) or (ascimin >= 64336 and
ascimin <= 69246) or
          (ascimin >= 124464 and ascimin <= 126705))
    then value = 'Arabic';
  elseif (ascimin >= 2304 and ascimin <= 3572)
    then value = 'Indian';
  elseif (ascimin >= 3585 and ascimin <= 3675)
    then value = 'Thai';
  elseif (ascimin >= 4096 and ascimin <= 4255)
    then value = 'Burmese';
  elseif ((ascimin >= 4352 and ascimin <= 4607) or (ascimin >= 12593 and
ascimin <= 12686) or
          (ascimin >= 12800 and ascimin <= 12926) or (ascimin >= 43360 and
ascimin <= 55203) or
          (ascimin >= 43360 and ascimin <= 55291) or (ascimin >= 65440 and
ascimin <= 65500))
    then value = 'Korean';
  elseif (ascimin >= 6016 and ascimin <= 6137)
```

```
    then value = 'Khmer';
  elseif ((ascimin >= 11904 and ascimin <= 12333) or (ascimin >= 12344 and
ascimin <= 12347) or
          (ascimin >= 13312 and ascimin <= 42182))
    then value = 'Chinese';
  elseif ((ascimin >= 12353 and ascimin <= 12543) or (ascimin >= 12784 and
ascimin <= 12799) or
          (ascimin >= 13008 and ascimin <= 13143))
    then value = 'Japanese';
  else value = 'Other';
  end if;
  return value;
end;
$$;
```

Solution2:

```
create or replace function script(title varchar)
  returns varchar as $script$
declare   max_ascii       int;
  declare min_ascii_gt127 int;
begin
  -- get the greatest code point in title
  select max(char_ascii)
  into max_ascii
  from
    (select ascii(chars) char_ascii
     from unnest(string_to_array(title, null)) chars) char_asciis;

  -- get the smallest code point that over 127 in title
  select min(char_ascii)
  into min_ascii_gt127
  from
    (select ascii(chars) char_ascii
     from unnest(string_to_array(title, null)) chars) char_asciis
  where char_ascii > 127;

  -- if the greatest code point is Latin, then the string uses the Latin script
  if max_ascii <= 740
     or max_ascii between 7424 and 8594
     or max_ascii between 11360 and 11391
     or max_ascii between 42786 and 43876
  then
    return 'Latin';
  else
    -- otherwise that the smallest code point over 127 in the string probably
defines the script
    return case
           when min_ascii_gt127 <= 740
```

```
                 or min_ascii_gt127 between 7424 and 8594
                 or min_ascii_gt127 between 11360 and 11391
                 or min_ascii_gt127 between 42786 and 43876
              then 'Latin'
            when min_ascii_gt127 between 880 and 1023
                 or min_ascii_gt127 between 7462 and 8446
              then 'Greek'
            when min_ascii_gt127 between 1024 and 1327
                 or min_ascii_gt127 between 7296 and 7544
                 or min_ascii_gt127 between 42560 and 42655
              then 'Cyrillic'
            when min_ascii_gt127 between 1536 and 2303
                 or min_ascii_gt127 between 64336 and 69246
                 or min_ascii_gt127 between 124464 and 126705
              then 'Arabic'
            when min_ascii_gt127 between 2304 and 3572
              then 'Indian'
            when min_ascii_gt127 between 3585 and 3675
              then 'Thai'
            when min_ascii_gt127 between 4096 and 4255
              then 'Burmese'
            when min_ascii_gt127 between 4352 and 4607
                 or min_ascii_gt127 between 12593 and 12686
                 or min_ascii_gt127 between 12800 and 12926
                 or min_ascii_gt127 between 43360 and 55203
                 or min_ascii_gt127 between 43360 and 55291
                 or min_ascii_gt127 between 65440 and 65500
              then 'Korean'
            when min_ascii_gt127 between 6016 and 6137
              then 'Khmer'
            when min_ascii_gt127 between 11904 and 12333
                 or min_ascii_gt127 between 12344 and 12347
                 or min_ascii_gt127 between 13312 and 42182
              then 'Chinese'
            when min_ascii_gt127 between 12353 and 12543
                 or min_ascii_gt127 between 12784 and 12799
                 or min_ascii_gt127 between 13008 and 13143
              then 'Japanese'
            else 'Other'
            end;
  end if;
end;
$script$
language plpgsql;
```