Function in SQL

PostgreSQL

Ahmad Yoosofan

Database course

University of Kashan, Spring 2021

sample 1

```
CREATE FUNCTION f_typetest01(INTEGER)
RETURNS bool
AS
'SELECT TRUE'
LANGUAGE sql;
```

```
1 CREATE OR REPLACE FUNCTION f_typetest01(INTEGE
2   RETURNS bool
3   AS
4   'SELECT TRUE'
5   LANGUAGE sql;
```

```
select f_typetest01(12);
```

```
 f_typetest01
--------------
 t
(1 row)
```

```
select *
from f_typetest01(12)
;
```

```
 f_typetest01
--------------
 t
(1 row)
```

Sample 2

```
1 CREATE OR REPLACE FUNCTION myf(d1 VARCHAR) RETURNS INTEGER
2 AS
3 $$
4 SELECT MAX(qty) FROM sp WHERE sn=d1;
5 $$ LANGUAGE SQL;
```

```
select myf('S1')
;
```

```
 myf
-----
 400
(1 row)
```

Sample 3

```
 1 DROP FUNCTION IF EXISTS test03();
 2 CREATE FUNCTION test03() RETURNS INTEGER AS $ABC$
 3 DECLARE
 4   quantity integer := 30;
 5   q2 integer := 30;
 6 BEGIN
 7   -- RAISE NOTICE 'Quantity here is %', quantity; -- Prints 30
 8   SELECT max(qty) into q2 from sp;
 9   RETURN q2;
10 END;
11 $ABC$ LANGUAGE plpgsql;
```

Function parts

```
 1 DROP FUNCTION IF EXISTS somefunc(VARCHAR(10));
 2
 3 CREATE FUNCTION somefunc(nm1 VARCHAR(10)) RETURNS INTEGER AS $ABC$
 4 DECLARE
 5   quantity integer := 30;
 6 BEGIN
 7   -- RAISE NOTICE 'Quantity here is %', quantity; -- Prints 30
 8   SELECT status INTO quantity FROM s WHERE sn=nm1;
 9   quantity := quantity * 50;
10   /*
11   -- Create a subblock
12   --
13   RAISE NOTICE 'Quantity here is %', quantity;*/
14   RETURN quantity;
15 END;
16 $ABC$ LANGUAGE plpgsql;
```

```
select somefunc('S1')
;
```

```
somefunc
----------
1000
(1 row)
```

```
 1 DROP FUNCTION IF EXISTS somefunc2(VARCHAR(10),nm2 VARCHAR(10));
 2
 3 CREATE FUNCTION somefunc2(nm1 VARCHAR(10), nm2 VARCHAR(10))
 4 RETURNS INTEGER AS $$
 5 DECLARE
 6   quantity integer := 30;
 7 BEGIN
 8   SELECT qty INTO quantity FROM sp WHERE sn=nm1 and pn=nm2;
 9   RAISE NOTICE 'Quantity here is %', quantity;
10   RETURN quantity;
11 END;
12 $$ LANGUAGE plpgsql;
```

```
 1 CREATE OR REPLACE FUNCTION somefunc03(nm1 CHAR(10), nm2 CHAR(10),
 2    o1 OUT INTEGER)
 3  AS $$
 4 DECLARE
 5   quantity integer := 30;
 6 BEGIN
 7   SELECT qty INTO quantity FROM sp WHERE sn=nm1 and pn=nm2;
 8   -- RAISE NOTICE 'Quantity here is %', quantity;
 9   o1 := quantity;
10 END;
11 $$ LANGUAGE plpgsql;
```

```
 1 CREATE OR REPLACE FUNCTION test03() RETURNS INTEGER AS $ABC$
 2 DECLARE
 3   quantity INTEGER := 30;
 4   q2  INTEGER  := 30;
 5   snp CHAR(10) := 'S1';
 6   pnp CHAR(10) := 'P1';
 7 BEGIN
 8   -- RAISE NOTICE 'Quantity here is %', quantity; -- Prints 30
 9   SELECT somefunc03(snp, pnp, q2);
10   RETURN q2;
11 END;
12 $ABC$ LANGUAGE plpgsql;
```

```
 1 DROP FUNCTION somefunc04(nm1 VARCHAR(10)) ;
 2 CREATE FUNCTION somefunc04(nm1 VARCHAR(10))
 3 RETURNS SETOF sp AS $$
 4 BEGIN
 5   IF nm1='S1' THEN
 6     RETURN QUERY SELECT * FROM sp WHERE sn='S2';
 7   ELSE
 8     RETURN QUERY SELECT * FROM sp WHERE sn=nm1 ;
 9   END IF;
10 END;
11 $$ LANGUAGE plpgsql;
```

```
 1 DROP FUNCTION somefunc05(nm1 VARCHAR(10),nm2 VARCHAR(10));
 2 CREATE FUNCTION somefunc05(nm1 VARCHAR(10),nm2 VARCHAR(10))
 3 RETURNS SETOF sp AS $$
 4 DECLARE
 5   i INTEGER :=0;
 6 BEGIN
 7   LOOP
 8     CASE i
 9       WHEN 1 THEN
10         RETURN QUERY SELECT * from sp where sn='S2';
11       WHEN 2 THEN
12         RETURN QUERY SELECT * FROM sp WHERE sn='S3';
13       ELSE
14         RETURN QUERY (SELECT * FROM sp WHERE pn=nm2);
15     END CASE;
16     i:=i+1;
17     EXIT WHEN i>2;
18   END LOOP;
19 END;
20 $$ LANGUAGE plpgsql;
```

```
select somefunc05('S1', 'P1');
```

```
somefunc05
-----------------------------------
 ("S1        ","P1        ",300)
 ("S2        ","P1        ",300)
 ("S2        ","P1        ",300)
 ("S2        ","P2        ",400)
 ("S3        ","P2        ",200)
(5 rows)
```

```sql
DROP FUNCTION somefunc05_3(nm2 VARCHAR(10));
CREATE FUNCTION somefunc05_3(nm2 VARCHAR(10))
RETURNS SETOF sp AS $$
DECLARE
  i INTEGER :=0;
BEGIN
  --RETURN QUERY SELECT * from sp where sn='S2';
  RETURN QUERY SELECT * FROM sp WHERE sn='S3';
END;
$$ LANGUAGE plpgsql;
```

```
sp=# select somefunc05('S1','P1');

            somefunc05
-----------------------------------
 ("S1         ","P1         ",300)
 ("S2         ","P1         ",300)
 ("S2         ","P1         ",300)
 ("S2         ","P2         ",400)
 ("S3         ","P2         ",200)
(5 rows)
```

```
 1 DROP FUNCTION somefunc05_4(nm2 VARCHAR(10));
 2 CREATE FUNCTION somefunc05_4(nm2 VARCHAR(10))
 3 RETURNS SETOF sp AS $$
 4 DECLARE
 5   i INTEGER :=0;
 6 BEGIN
 7   --RETURN QUERY SELECT * from sp where sn='S2';
 8   --RETURN QUERY SELECT * FROM sp WHERE sn='S3';
 9   RETURN QUERY (SELECT * FROM sp WHERE pn=nm2);
10 END;
11 $$ LANGUAGE plpgsql;
```

```
sp=# select somefunc05('S1','P1');
            somefunc05
-----------------------------------
 ("S1         ","P1         ",300)
 ("S2         ","P1         ",300)
 ("S2         ","P1         ",300)
 ("S2         ","P2         ",400)
 ("S3         ","P2         ",200)
(5 rows)
```

```
 1 DROP FUNCTION somefunc05_5(nm2 VARCHAR(10));
 2 CREATE FUNCTION somefunc05_5(nm2 VARCHAR(10))
 3 RETURNS SETOF sp AS $$
 4 DECLARE
 5   i INTEGER :=0;
 6 BEGIN
 7   RETURN QUERY SELECT * from sp where sn='S2';
 8   RETURN QUERY SELECT * FROM sp WHERE sn='S3';
 9   RETURN QUERY SELECT * FROM sp WHERE pn=nm2;
10 END;
11 $$ LANGUAGE plpgsql;
```

```
sp=# select somefunc05('S1','P1');
            somefunc05
-----------------------------------
 ("S1         ","P1         ",300)
 ("S2         ","P1         ",300)
 ("S2         ","P1         ",300)
 ("S2         ","P2         ",400)
 ("S3         ","P2         ",200)
(5 rows)
```

```
 1 DROP FUNCTION sf06() ;
 2 CREATE FUNCTION sf06()
 3 RETURNS SETOF sp AS $$
 4 DECLARE
 5   i INTEGER :=0;
 6   r sp%rowtype;
 7 BEGIN
 8   FOR r IN SELECT * FROM sp LOOP
 9     CASE i
10       WHEN 1 THEN
11         RETURN NEXT r;
12       WHEN 2 THEN
13         RETURN NEXT r;
14       ELSE
15         RETURN NEXT r;
16     END CASE;
17     i:=i+1;
18     EXIT WHEN i>2;
19   END LOOP;
20 END;
21 $$ LANGUAGE plpgsql;
```

```
sp=# \i simple.function.06.sql
CREATE FUNCTION
sp=# select * from sf06('S1','P1');
    sno     |    pno     | qty
------------+------------+-----
 S1         | P1         | 300
 S1         | P2         | 200
 S1         | P3         | 400
(3 rows)
```

Simple Functions 1

```
1 CREATE OR REPLACE
2 FUNCTION get_year(INTEGER)
3 RETURNS INTEGER AS $$
4 BEGIN
5   RETURN ( $1 / 10000);
6 END;
7 $$ LANGUAGE plpgsql
8 RETURNS NULL ON NULL INPUT;
```

```
1 CREATE OR REPLACE
2 FUNCTION get_month(INTEGER)
3 RETURNS INTEGER AS $$
4 BEGIN
5   RETURN ( $1 / 100 ) % 100;
6 END;
7 $$ LANGUAGE plpgsql
8 RETURNS NULL ON NULL INPUT;
```

```
1 CREATE OR REPLACE
2 FUNCTION get_day(INTEGER)
3 RETURNS INTEGER AS $$
4 BEGIN
5   RETURN $1  % 100 ;
6 END;
7 $$ LANGUAGE plpgsql
8 RETURNS NULL ON NULL INPUT;
```

```
1 CREATE OR REPLACE FUNCTION
2 get_year_and_month(INTEGER)
3 RETURNS INTEGER AS $$
4 BEGIN
5   RETURN ( $1 / 100);
6 END;
7 $$ LANGUAGE plpgsql
8 RETURNS NULL ON NULL INPUT;
```

```
CREATE OR REPLACE FUNCTION convert_int_date_to_varchar10(integer)
RETURNS VARCHAR(10) AS $BODY$
BEGIN
  RETURN ( substring($1::text FROM 1 FOR 4) || '-' ||
    substring($1::text FROM 5 FOR 2) || '-' ||
    substring($1::text FROM 7 FOR 2) )::varchar(10);
END;
$BODY$ LANGUAGE plpgsql
RETURNS NULL ON NULL INPUT;
```

```
 1 CREATE OR REPLACE FUNCTION uniform_text_jdatei4search(text)
 2 RETURNS VARCHAR(10) AS $BODY$
 3 DECLARE
 4   lmy_str1 varchar(10);
 5 BEGIN
 6   lmy_str1 := substring($1 from '^[0-9]+') ;
 7   IF character_length(lmy_str1)=8 THEN
 8     RETURN (substring(lmy_str1 FROM 1 FOR 4) || '-' ||
 9         substring(lmy_str1 FROM 5 FOR 2) || '-' ||
10         substring(lmy_str1 FROM 7 FOR 2) )::varchar(10);
11   ELSEIF character_length(lmy_str1)=6 THEN
12     RETURN (substring(lmy_str1 FROM 1 FOR 4) || '-' ||
13         substring(lmy_str1 FROM 5 FOR 2) )::varchar(10);
14   END IF;
15 END;
16 $BODY$ LANGUAGE plpgsql
17 RETURNS NULL ON NULL INPUT;
```

```
 1 CREATE OR REPLACE FUNCTION composit_idate(text,integer)
 2 RETURNS TEXT AS $BODY$
 3 DECLARE
 4   lmy_astr1 TEXT ARRAY;
 5 BEGIN
 6   lmy_astr1 := regexp_split_to_array($1 ,'\$\@') ;
 7   --RAISE NOTICE 'values are %', lmy_astr1[1];
 8   RETURN lmy_astr1[$2];
 9 END;
10 $BODY$ LANGUAGE plpgsql
11 RETURNS NULL ON NULL INPUT;
```

```
 1 CREATE OR REPLACE FUNCTION is_leap_year(INTEGER)
 2 RETURNS INTEGER AS $$
 3 DECLARE
 4   p INTEGER;
 5   leap INTEGER;
 6 BEGIN
 7   p := ( ( $1 + 2346 ) % 2820 ) % 128 ;
 8   IF p = 5 OR p = 9 OR p = 13 OR p = 17 OR p = 21 OR p = 25 OR p = 29 OR
 9      p = 34 OR p = 38 OR p = 42  OR p = 46 OR p = 50 OR p = 54 OR p = 58 OR
10      p = 62 OR p = 67 OR p = 71 OR p = 75  OR p = 79 OR p = 83 OR
11      p = 87 OR p = 91 OR p = 95 OR p = 100 OR p = 104 OR p = 108 OR
12      p = 112 OR p = 116 OR p = 120 OR p = 124 THEN
13     leap := 1;
14   ELSE
15     leap := 0;
16   END IF;
17   RETURN leap ;
18 END;
19 $$ LANGUAGE plpgsql
20 RETURNS NULL ON NULL INPUT;
```

Related

- http://www.postgresql.org/docs/9.5/static/sql-createfunction.html
- http://stackoverflow.com/questions/30782925/postgresql-how-to-drop-function-if-exists-without-specifying-parameters
- http://stackoverflow.com/questions/30782925/postgresql-how-to-drop-function-if-exists-without-specifying-parameters
- https://github.com/malimome/pgsql-jalalical/blob/master/install/pdate.source
- https://gist.github.com/ilius
- https://stackoverflow.com/questions/52436973/postgresql-does-postgresql-support-persian-calendar