1. 编写一个函数，函数名为get\_student\_phone，无接收参数，返回一个随机的手机号，长度11位，手机号以'159'或 '137'开头，要求任意满足该要求的手机号能等概率生成。

SQL语句：

create or replace function get\_student\_phone() returns character varying

as $$

declare

result varchar(50);

num integer;

begin

select array\_to\_string(array(select chr((48 + round(random() \* 9)) :: integer) from generate\_series(1,8)), '') into result;

num = random() \* 1;

if num = 1 then

result = '137' || result;

else

result = '159' || result;

end if;

return result;

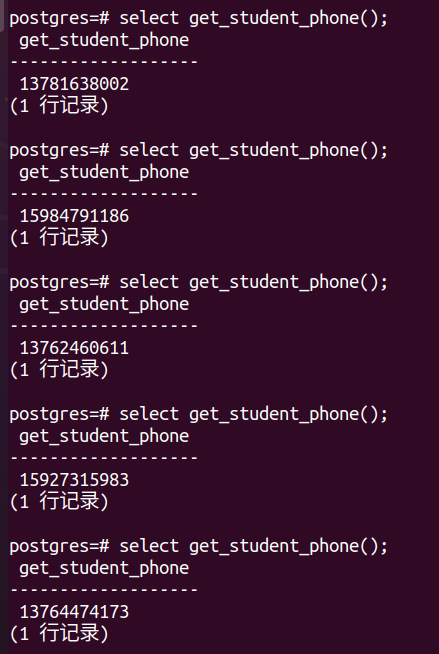
end;

$$

language plpgsql;

百分之五十的概率生成137或者159开头的电话号码，后面8位电话号码随机等概率生成

查询结果：



2. 编写一个函数，函数名为get\_student\_date，无接收参数，返回一个随机的日期，日期格式为'YYYY-MMDD'。要求返回的日期区间为[2020-01-01, 2021-12-31]，其中，要求生成2020年份概率为60%，生成2021年份概率为40%，此外，月和日则是等概率返回。

SQL语句：

create or replace function get\_student\_date() returns character varying

as $$

declare

result varchar(100);

month integer;

year integer;

day integer;

num integer;

begin

num = random() \* 9;

if num >= 6 then

year = 2021;

result = '2021-';

else

year = 2020;

result = '2020-';

end if;

month = random() \* 12 + 1;

if month < 10 then

result = result || '0' || cast(month as varchar) || '-';

else

result = result || cast(month as varchar) || '-';

end if;

if month = 1 or month = 3 or month = 5 or month = 7 or month = 8 or month = 10 or month = 12 then

day = random() \* 31 + 1;

end if;

if month = 4 or month = 6 or month = 9 or month = 11 then

day = random() \* 30 + 1;

end if;

if month = 2 and year = 2020 then

day = random() \* 29 + 1;

end if;

if month = 2 and year = 2021 then

day = random() \* 28 + 1;

end if;

if day < 10 then

result = result || '0' || cast(day as varchar) ;

else

result = result || cast(day as varchar);

end if;

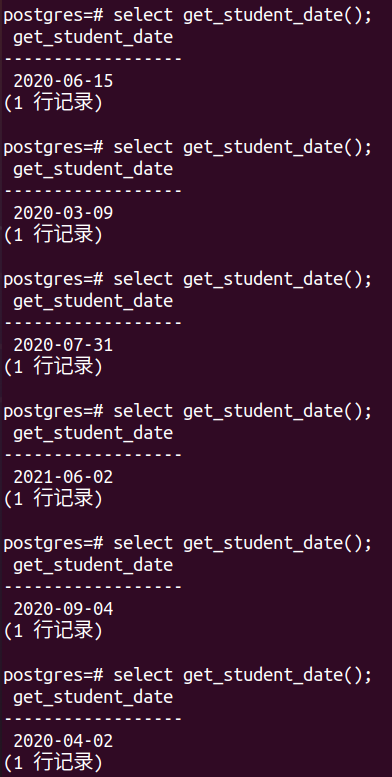
return result;

end;

$$

language plpgsql;

查询结果：



3. 编写一个函数，函数名为create\_student\_table，无接收参数。在该函数中，新建一个数据表 student，该数据表拥有3个字段，分别是 student\_id, phone\_num, enrollment\_date，其中 student\_id 为自增的序列，从1开始自增，且为主键；然后，往该数据表新增 15条记录，这 15条记录中，phone\_num和 enrollment\_date分别使用上述自己编写的第一个和第二个函数生成。最后返回该表。该函数理应可以连续调用多次，每次生成并返回的表都不一样。

SQL语句：

create or replace function create\_student\_table()

returns table(student\_id integer, phone\_num varchar, enrollment\_date varchar)

as $$

declare

num integer := 1;

begin

drop table if exists student;

create table student(student\_id integer, phone\_num varchar, enrollment\_date varchar);

while num <= 15 loop

insert into student(student\_id, phone\_num, enrollment\_date)

values(num,get\_student\_phone(),get\_student\_date());

num = num + 1;

end loop;

return query select \* from student;

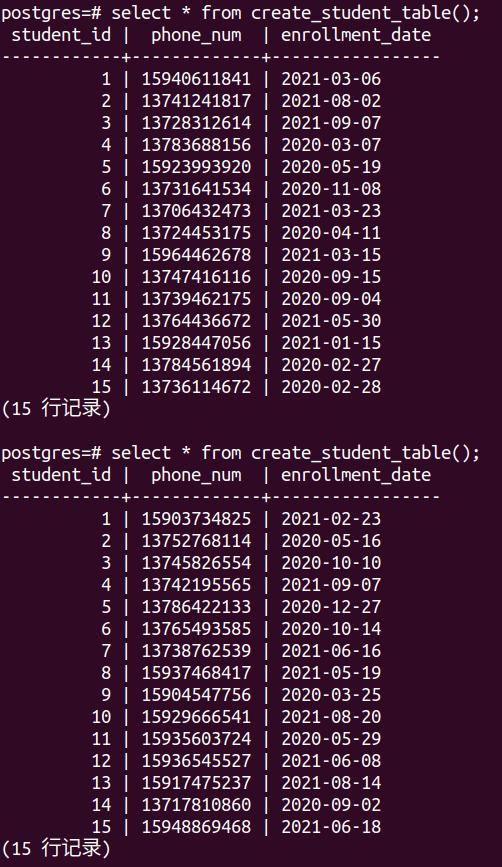
end;

$$

language plpgsql;

创建一个表格student，通过调用get\_student\_phone(),get\_student\_date()不断地往表格中插入数据

查询结果：



4. 查询：使用 student表，找出所有enrollment\_date在2020年7月1日（包括这一天）之后的学生，并输出其phone\_num。

SQL语句：

create or replace function select\_student\_by\_date()

returns table(student integer,phone varchar)

as $$

begin

return query select student\_id, phone\_num from student where enrollment\_date >= '2020-07-01';

end;

$$

language plpgsql;

通过比较字符串大小，即可获得日期大于等于2020-07-01的学生

查询结果：

