第一次作业

1. 找出销售税小于0.15的销售区域，显示这些销售区域的名字（不输出重复的名字）。

```
select distinct d_name
from bmsql_district
where d_tax < 0.15;
```

2. 找出给state HS（销售区域）供货的仓库都来自哪个state和city。

```
select w_state , w_city
from bmsql_warehouse , bmsql_district
where d_state = 'HS' and w_id = d_w_id;
```

3. 找出在某个仓库中货物数量少于18而且价格为80的货物和对应的仓库，输出这些货物的ID、对应仓库的ID和货物的剩余数量。（提示：在STOCK表和ITEM表中查询）

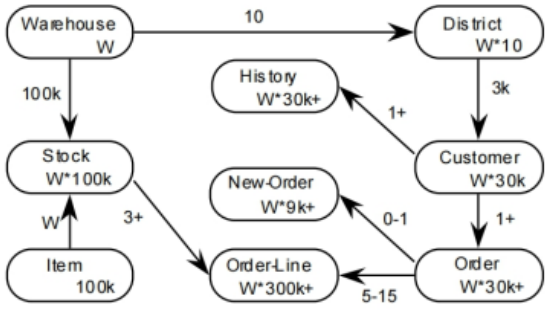
```
select s_i_id , s_w_id , s_quantity
from bmsql_stock，bmsql_item
where s_i_id = i_id and i_price = 80 and s_quantity < 18;
```

4. 找出满足以下要求的仓库的ID和名字（不输出重复的ID和名字）：有来自福建省（state为FJ）且享有八折优惠的顾客购买过该仓库的商品。

a、customer表，order表和history表都可以查询购买记录。但是需要注意，order/history和customer的对应关系为：

o_w_id=c_w_id and o_d_id=c_d_id and o_c_id=c_id

h_c_w_id=c_w_id and h_c_d_id=c_d_id and h_c_id=c_id



P16： (O_W_ID, O_D_ID, O_C_ID) Foreign Key, references (C_W_ID, C_D_ID, C_ID)

P15： (H_C_W_ID, H_C_D_ID, H_C_ID) Foreign Key, references (C_W_ID, C_D_ID, C_ID)

b、八折指的应该是discount=0.2　　sum(OL_AMOUNT) * (1 - C_DISCOUNT) * (1 + W_TAX + D_TAX)

正确样例：

```
select distinct w_id,w_name
from bmsql_warehouse,bmsql_customer
where w_id = c_w_id and c_state = 'FJ' and c_discount = 0.2;
```

```
select distinct w_id, w_name
from bmsql_warehouse, bmsql_customer, bmsql_history
where  c_state = 'FJ' and c_discount = 0.2 and h_w_id = w_id and h_c_id = c_id and h_c_w_id = c_w_id
and h_c_d_id = c_d_id;
```

```
select distinct w_id, w_name
from bmsql_warehouse, bmsql_customer, bmsql_oorder
where  c_state = 'FJ' and c_discount = 0.2 and o_w_id = w_id and o_w_id=c_w_id and o_d_id=c_d_id
and o_c_id=c_id;
```

5. 找出享有七折优惠而且信用良好，同时在state UV或HS（销售区域）有购买商品的顾客，显示他们的姓名（包括中间名）。

a、如果使用order表或者history表查看购买记录需要注意映射关系

b、销售区域应该是d_state而不是c_state

c、七折指的应该是discount=0.3

```
select distinct c_first , c_middle , c_last
from bmsql_customer,bmsql_district
where c_d_id = d_id and c_discount = 0.3 and c_credit = 'GC' and (d_state = 'UV' or d_state = 'HS');
```

以order表为例：

```
select distinct c_first,c_middle, c_last
from bmsql_customer, bmsql_oorder, bmsql_district
where c_discount = 0.3 and c_credit = 'GC' and (d_state = 'UV' or d_state = 'HS') and o_d_id = d_id
and o_c_id=c_id and o_w_id=c_w_id and o_d_id=c_d_id;
```

第二次作业

一、统计函数的使用：

1、在表item中计算所有商品的数量，价格平均值，价格最大值，价格最小值，价格方差。

```
select count(i_price), avg(i_price), max(i_price), min(i_price), variance(i_price)
from bmsql_item;
```

2、在表stock中统计每个仓库保存的商品数量平均值，输出列为w_id, avg。

```
select s_w_id as w_id,avg(s_quantity) as avg from bmsql_stock group by s_w_id;
```

二、正则表达式的使用：

1、找出所有以'NB'为名字开头或者以‘ VT ‘为结尾的商品的所有信息；（用一个正则表达式解决，不要用or）

```
select * from bmsql_item where i_name ~'^NB|VT$';
```

2、统计以名字开头字母在h-m之间的商品数量，以及平均价格。

```
select count(i_price),avg(i_price) from bmsql_item where i_name ~'^[h-m]';
```

三、all/any的使用：

1、对于二.1中的商品，找出在所有仓库中都有储备的商品，输出商品的所有信息。

```
select * from bmsql_item
where i_name ~'^NB|VT$' and
10 < all(select s_quantity from bmsql_stock where s_i_id=i_id);
```

2、找出至少有一个仓库主要储备且该仓库销售税（w_tax）大于0.16的商品（使用any），输出这些商品的所有信息。

```
select * from bmsql_item
where 95< any(select s_quantity from bmsql_stock,bmsql_warehouse where s_i_id=i_id and
s_w_id=w_id and w_tax>0.16);
```

四、嵌套查询(in)：

1、找到有商品税大于0.18的仓库主要储备的所有商品，输出它们的所有信息。

找到商品税大于0.18的仓库：select w_id from bmsql_warehouse where w_tax>0.18

同时满足商品税大于0.18和储备量>95两个条件：

```
select * from bmsql_item
where 95 < any(select s_quantity from bmsql_stock where s_i_id=i_id and
s_w_id in (select w_id from bmsql_warehouse where w_tax>0.18) );
```

2、找到主要贮备有以'SP'为开头的商品的仓库，输出仓库的所有信息。

找到以'SP'为开头的商品：select i_id from bmsql_item where i_name ~'^SP'

同时满足'SP'为开头大于0.18和储备量>95两个条件

```
select * from bmsql_warehouse
where 95 < any(select s_quantity from bmsql_stock where s_w_id=w_id and
s_i_id in (select i_id from bmsql_item where i_name ~'^SP') );
```

五、综合题：

1、找到以'SP'为开头，且在所有仓库储存的平均数量大于50的商品的全部信息。

计算i_id商品的仓库储存平均数量：
idselect avg(s_quantity) from bmsql_stock where s_i_id=i_id

用all或any（因为括号内只有一个值）判断平均数量大于50

```
select * from bmsql_item
where i_name ~'^SP' and
50 < any(select avg(s_quantity) from bmsql_stock where s_i_id=i_id );
```

2、找到所有满足条件的仓库的编号(w_id)：该仓库在所有地区的销售税都小于0.15。

```
select w_id from bmsql_warehouse
where true = all(select d_tax<0.15 from bmsql_district where d_w_id=w_id);
```

3、统计五.2中的仓库主要储备的商品数量，价格平均值，输出列为w_id, number，avg_price。

使用五.2的命令收集仓库id，然后在对item表格进行筛选和统计：

```
select s_w_id as w_id,avg(i_price) as avg_price ,count(i_id) as number from bmsql_item,bmsql_stock
where i_id=s_i_id and s_quantity>95 and s_w_id in
(select w_id from bmsql_warehouse
where true = all(select d_tax<0.15 from bmsql_district where d_w_id=w_id) )
group by s_w_id;
```

第三次作业

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。

   示例：

```
select get_student_phone();
get_student_phone
------------------
13790119007
```

```
CREATE or replace function get_student_phone() returns bigint as $$
DECLARE
    latter bigint ;
    former bigint;
BEGIN
    latter := trunc(random()*100000000);
    if random() > 0.5 then
        former := 13700000000;
    else
        former := 15900000000;
    end if;

    RETURN former+latter;
END;
$$ language plpgsql;
```

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-MM-DD'。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。

   示例：

```
select get_student_date();
get_student_date
---------------
2020-12-13
```

```
CREATE or replace function get_student_date() returns date as $$
DECLARE
    days int;
    dates date;
    first_date date;
BEGIN
    first_date := '2020-01-01';
    if random() > 0.4  then
        days := trunc( random()*366 );
        dates := first_date + days;
    else
        days := trunc( random()*365 );
        dates := first_date + 366 + days;
    end if;

    RETURN dates;
END;
$$ language plpgsql;
```

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。

   示例：

```
select * from create_student_table();
student_id | phone_num    | enrollment_date
-----------------------------------
    1      | 13790829207  | 2020-02-25
...
    15     | 15923624934  | 2021-04-15
```

```
CREATE or replace function create_student_table() returns table(student_id int , phone_num bigint , enrollment_date date) as $$
    DECLARE
    BEGIN
        drop table if exists student;
        create table student(student_id int not null, phone_num bigint not null, enrollment_date date not null, primary key(student_id) );

        for i in 1..15 loop
            insert into student values(i , get_student_phone() , get_student_date());
        end loop;
        return query select* from student;
END;
$$ language plpgsql;
```

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
select phone_num
from student
where enrollment_date >= '2020-07-01';
```