数据库系统课程实验报告

实验名称:	实验三：数据基本查询
实验日期:	2022/4/7
实验地点:	厦门大学德旺图书馆
提交日期:	2022/4/7

学号:	20420192201952
姓名:	庾晓萍
专业年级:	软工 2020 级
学年学期:	2021-2022 学年第二学期

1. 实验目的

- 熟练掌握 openGauss 单表查询的语法结构及其使用方法
- 掌握设计正确查询语句以实现查询要求的方法
- -普通单表查询、模糊查询、聚集函数、分组统计和排序
- 掌握 Group by 的使用
- 正确区分元组过滤条件（WHERE 子句）和分组过滤条件（HAVING 短语）的异同
- 掌握 Order by 的使用
- 掌握使用 DISTINCT 实现查询结果的去重方法
- 掌握空值 NULL 的使用方法
- 掌握查询过程中别名的使用方法
- 掌握自身连接的使用方法

2. 实验内容和步骤

（1）查询顾客表中的顾客号（customer_id）、顾客名（name）和信用卡额度（credit_limit）

```
sale=> SELECT customer_id,name,credit_limit FROM customers;
 customer_id |                 name                 | credit_li
mit
-------------+--------------------------------------+----------
----
         177 | United Continental Holdings          |      5000
.00
         180 | INTL FCStone                         |      5000
.00
         184 | Publix Super Markets                 |      1200
.00
         187 | ConocoPhillips                       |      2400
.00
         190 | 3M                                   |      1200
.00
         192 | Exelon                               |       500
.00
         208 | Tesoro                               |       500
.00
         207 | Northwestern Mutual                  |      3600
.00
         200 | Enterprise Products Partners         |      2400
.00
         204 | Rite Aid                             |      3600
--More--
```

（2）查询顾客的所有信息，使用 SELECT * FROM customers;查询表的数据。

```
>_ root@114.116.235.207 ×
 customer_id |              name              |                  address                  |              website               | credit_lim
it
-------------+--------------------------------+-------------------------------------------+------------------------------------+-----------
---
         177 | United Continental Holdings    | 2904 S Salina St, Syracuse, NY            | http://www.unitedcontinentalholdings.com |      5000.
00
         180 | INTL FCStone                   | 5344 Haverford Ave, Philadelphia, PA      | http://www.intlfcstone.com         |      5000.
00
         184 | Publix Super Markets           | 1795 Wu Meng, Muang Chonburi,             | http://www.publix.com              |      1200.
00
         187 | ConocoPhillips                 | Walpurgisstr 69, Munich,                  | http://www.conocophillips.com      |      2400.
00
         190 | 3M                             | Via Frenzy 6903, Roma,                    | http://www.3m.com                  |      1200.
00
         192 | Exelon                         | Via Luminosa 162, Firenze,                | http://www.exeloncorp.com          |       500.
00
         208 | Tesoro                         | Via Notoriosa 1942, Firenze,              | http://www.tsocorp.com             |       500.
00
         207 | Northwestern Mutual            | 1831 No Wong, Peking,                     | http://www.northwesternmutual.com  |      3600.
00
         200 | Enterprise Products Partners   | Via Notoriosa 1949, Firenze,              | http://www.enterpriseproducts.com  |      2400.
00
         204 | Rite Aid                       | Piazza Cacchiatore 23, San Giminiano,     | http://www.riteaid.com             |      3600.
00
         212 | Qualcomm                       | Piazza Svizzera, Milano,                  | http://www.qualcomm.com            |       500.
00
         216 | EMC                            | Via Delle Grazie 11, San Giminiano,       | http://www.emc.com                 |       700.
00
         220 | Time Warner Cable              | 1597 Legend St, Mysore, Kar               | http://www.twc.com                 |      3700.
00
         223 | Northrop Grumman               | 1606 Sangam Blvd, New Delhi,              | http://www.northropgrumman.com     |      5000.
00
          39 | Lear                           | 2115 N Towne Ln Ne, Cedar Rapids, IA      | http://www.lear.com                |       500.
00
          43 | Facebook                       | 5112 Sw 9Th St, Des Moines, IA            | http://www.facebook.com            |       500.
00
          46 | Supervalu                      | 8989 N Port Washington Rd, Milwaukee, WI  | http://www.supervalu.com           |       500.
--More--
```

（3）查询订单表中的订单号，顾客号，状态，订单日期，并按订单日期降序显示结果。SELECT order_id,customer_id,status,order_date FROM orders ORDER BY order_date Desc;

order_id	customer_id	status	order_date
88	6	Shipped	2017-11-01 00:00:00
94	1	Shipped	2017-10-27 00:00:00
1	4	Pending	2017-10-15 00:00:00
14	48	Shipped	2017-09-28 00:00:00
15	49	Shipped	2017-09-27 00:00:00
17	17	Shipped	2017-09-27 00:00:00
36	51	Shipped	2017-09-05 00:00:00
57	68	Shipped	2017-08-24 00:00:00
28	6	Canceled	2017-08-15 00:00:00
29	44	Shipped	2017-08-14 00:00:00
30	45	Shipped	2017-08-12 00:00:00
31	46	Canceled	2017-08-12 00:00:00
60	1	Shipped	2017-06-30 00:00:00
20	20	Shipped	2017-05-27 00:00:00
21	21	Pending	2017-05-27 00:00:00
40	55	Shipped	2017-05-11 00:00:00
41	9	Shipped	2017-05-11 00:00:00
3	5	Shipped	2017-04-26 00:00:00
5	5	Canceled	2017-04-09 00:00:00
98	48	Shipped	2017-03-18 00:00:00
69	44	Canceled	2017-03-17 00:00:00
32	47	Shipped	2017-03-09 00:00:00
33	48	Shipped	2017-03-07 00:00:00
71	46	Shipped	2017-02-21 00:00:00
70	45	Canceled	2017-02-21 00:00:00
46	58	Pending	2017-02-20 00:00:00
44	2	Pending	2017-02-20 00:00:00
45	57	Shipped	2017-02-20 00:00:00
37	52	Shipped	2017-02-19 00:00:00
7	7	Shipped	2017-02-15 00:00:00

（4）查询联系表中的名（first name）和姓（last name），并按名升序，姓降序显示。 SELECT first_name,last_name FROM contacts ORDER BY first_name Asc,last_name Desc;

first_name	last_name
Aaron	Holder
Adah	Myers
Adam	Jacobs
Adrienne	Lang
Agustina	Conner
Al	Schultz
Aleshia	Reese
Alessandra	Estrada
Alexandra	Mcgowan
Alvaro	Hooper
Alysa	Kane
Amado	Jefferson
Amber	Brady
Amira	Macdonald
Annabelle	Butler
Annelle	Lawrence
Annice	Boyer
Arlene	Elliott
Arlette	Thornton
Arlyne	Ingram
Audie	Flores
Audrea	Hayden
Audrie	Cannon
Aundrea	Barry
Aurea	Sanders
Avis	Moore
Barbie	Carter
Basilia	Downs
Beatrice	Ford
Bill	Stein
Birgit	Stephenson
Blanche	Robbins
Bobby	Wilson
Brandie	Buchanan
Brock	Webb

--More--

(5) 执行以下语句并观察 state 列 NULL 值的显示位置，得出结论。

```
SELECT country_id, city,state FROM locations ORDER BY city,state;

SELECT country_id, city,state FROM locations ORDER BY state ASC NULLS FIRST ;

SELECT country_id, city,state FROM locations ORDER BY state ASC NULLS LAST;
```

① 在 locations 表中选择出 country_id,city,state 三列，同时按照 city 和 state 排序。

```
sale=> SELECT country_id, city,state FROM locations ORDER BY city,state;
 country_id |        city         |       state
------------+---------------------+-------------------
 CN         | Beijing             |
 CH         | Bern                | BE
 IN         | Bombay              | Maharashtra
 CH         | Geneva              | Geneve
 JP         | Hiroshima           |
 UK         | London              |
 MX         | Mexico City         | Distrito Federal,
 DE         | Munich              | Bavaria
 UK         | Oxford              | Oxford
 IT         | Roma                |
 BR         | Sao Paulo           | Sao Paulo
 US         | Seattle             | Washington
 SG         | Singapore           |
 US         | South Brunswick     | New Jersey
 US         | South San Francisco | California
 US         | Southlake           | Texas
 UK         | Stretford           | Manchester
 AU         | Sydney              | New South Wales
 JP         | Tokyo               | Tokyo Prefecture
 CA         | Toronto             | Ontario
 NL         | Utrecht             | Utrecht
 IT         | Venice              |
 CA         | Whitehorse          | Yukon
(23 rows)
```

② 在 locations 表中选择出 country_id,city,state 三列，同时按照 state 升序排序，同时空格放在最前面。

```
sale=> SELECT country_id, city,state FROM locations ORDER BY state ASC NULLS FIRST ;
 country_id |        city         |       state
------------+---------------------+-------------------
 CN         | Beijing             |
 IT         | Venice              |
 IT         | Roma                |
 JP         | Hiroshima           |
 UK         | London              |
 SG         | Singapore           |
 CH         | Bern                | BE
 DE         | Munich              | Bavaria
 US         | South San Francisco | California
 MX         | Mexico City         | Distrito Federal,
 CH         | Geneva              | Geneve
 IN         | Bombay              | Maharashtra
 UK         | Stretford           | Manchester
 US         | South Brunswick     | New Jersey
 AU         | Sydney              | New South Wales
 CA         | Toronto             | Ontario
 UK         | Oxford              | Oxford
 BR         | Sao Paulo           | Sao Paulo
 US         | Southlake           | Texas
 JP         | Tokyo               | Tokyo Prefecture
 NL         | Utrecht             | Utrecht
 US         | Seattle             | Washington
 CA         | Whitehorse          | Yukon
(23 rows)
```

③ 在 locations 表中选择出 country_id,city,state 三列，同时按照 state 升

序排序，同时空格放在最后面。

```
sale=> SELECT country_id, city,state FROM locations ORDER BY state ASC NULLS LAST;
 country_id |        city         |       state
------------+---------------------+-------------------
 CH         | Bern                | BE
 DE         | Munich              | Bavaria
 US         | South San Francisco | California
 MX         | Mexico City         | Distrito Federal,
 CH         | Geneva              | Geneve
 IN         | Bombay              | Maharashtra
 UK         | Stretford           | Manchester
 US         | South Brunswick     | New Jersey
 AU         | Sydney              | New South Wales
 CA         | Toronto             | Ontario
 UK         | Oxford              | Oxford
 BR         | Sao Paulo           | Sao Paulo
 US         | Southlake           | Texas
 JP         | Tokyo               | Tokyo Prefecture
 NL         | Utrecht             | Utrecht
 US         | Seattle             | Washington
 CA         | Whitehorse          | Yukon
 IT         | Venice              |
 CN         | Beijing             |
 IT         | Roma                |
 SG         | Singapore           |
 UK         | London              |
 JP         | Hiroshima           |
(23 rows)
```

（6）查询订单细节表中（order_items）的产品号和数量，查询结果应无重复元组：

SELECT DISTINCT product_id,quantity FROM order_items;

```
sale=>  SELECT DISTINCT product_id,quantity FROM order_items;
 product_id | quantity
------------+----------
        126 |   105.00
         65 |    99.00
        216 |    82.00
        186 |   116.00
        121 |   120.00
         23 |   142.00
        266 |    83.00
        194 |    75.00
        110 |    94.00
        230 |    49.00
        188 |   130.00
        106 |   137.00
         34 |   126.00
        168 |   136.00
        165 |    46.00
         10 |   147.00
         29 |   133.00
        163 |    98.00
         13 |    75.00
         28 |    63.00
        208 |   139.00
        287 |    61.00
        278 |   139.00
         62 |   148.00
```

（7）查询产品表中的产品名为‘Kingston’的产品名，产品描述和价格：SELECT product_name,description,list_price FROM products WHERE product_name='Kingston';

```
sale=> SELECT product_name,description,list_price FROM products WHERE product_name='Kingston';
 product_name |                        description                         | list_price
--------------+------------------------------------------------------------+------------
 Kingston     | Speed:DDR4-2133,Type:288-pin DIMM,CAS:15Module:4x16GBSize:64GB |     741.63
 Kingston     | Speed:DDR3-1333,Type:240-pin DIMM,CAS:9Module:4x16GBSize:64GB  |     671.38
 Kingston     | Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x8GBSize:32GB  |     653.50
 Kingston     | Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x16GBSize:64GB |     644.00
(4 rows)
```

（8）查询产品表中所有价格大于 500 且 category_id 为 4 的产品名和价格：SELECT product_name,list_price FROM products WHERE category_id=4 AND list_price>500;

```
sale=> SELECT product_name,list_price FROM products WHERE category_id=4 AND list_price>500;
          product_name          | list_price
--------------------------------+------------
 Supermicro X10SDV-8C-TLN4F     |     948.99
 Intel DP35DPM                  |     789.79
 Asus X99-E-10G WS              |     649.00
 Asus ROG MAXIMUS IX EXTREME    |     573.99
 Asus RAMPAGE V EXTREME         |     572.96
 Asus Z10PE-D8 WS               |     561.59
 MSI X99A GODLIKE GAMING CARBON |     549.59
 Supermicro H8DG6-F             |     525.99
 Asus Rampage V Edition 10      |     519.99
 Gigabyte GA-Z270X-Gaming 9     |     503.98
(10 rows)
```

（9）查询产品表中所有价格在 650 和 680 之间的产品名和价格并按价格升序显示结果。SELECT product_name,list_price FROM products WHERE list_price between 650 AND 680 order by list_price asc;

```
sale=> SELECT product_name,list_price FROM products WHERE list_price between 650 AND 680 order by list_price asc;
       product_name        | list_price
---------------------------+------------
 Kingston                  |     653.50
 Corsair Dominator Platinum |     659.99
 Intel Core i7-3930K       |     660.00
 Kingston                  |     671.38
 G.Skill Ripjaws V Series  |     677.99
 Intel Core i7-7820X       |     678.75
(6 rows)
```

（10）查询雇员表中的名和姓，名和姓的字段分别显示为"First Name"和"Family Name": SELECT first_name "First Name",last_name "Family Name" FROM employees;

First Name	Family Name
Tommy	Bailey
Blake	Cooper
Jude	Rivera
Mohammad	Peterson
Harper	Spencer
Louie	Richardson
Nathan	Cox
Bobby	Torres
Charles	Ward
Gabriel	Howard
Emma	Perkins
Amelie	Hudson
Gracie	Gardner
Rory	Kelly
Isabella	Cole
Jessica	Woods
Ella	Wallace
Ava	Sullivan
Mia	West
Evie	Harrison
Scarlett	Gibson
Ruby	Mcdonald
Chloe	Cruz
Isabelle	Marshall
Daisy	Ortiz
Freya	Gomez
Elizabeth	Dixon
Florence	Freeman
Alice	Wells
Charlotte	Webb
Sienna	Simpson
Matilda	Stevens
Evelyn	Tucker
Eva	Porter
Millie	Hunter

--More--

（11）查询产品表中的产品名及毛利，并按毛利结果降序显示，毛利名为 gross_profit，毛利=list_price - standard_cost：

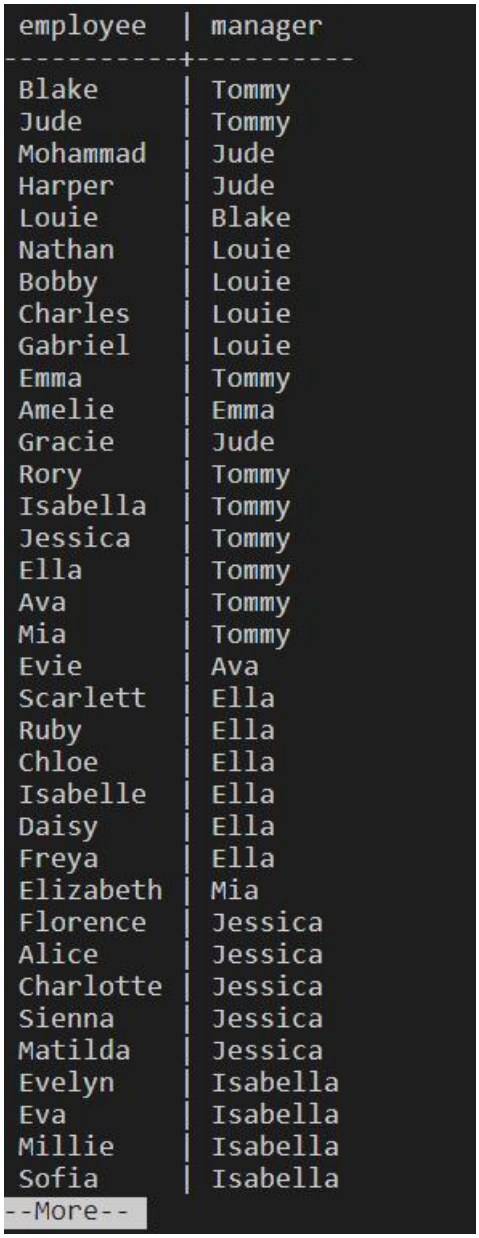
SELECT product_name,list_price - standard_cost "gross_profit" FROM products order by gross_profit DESC;

```
sale=> SELECT product_name,list_price - standard_cost "gross_profit" FROM products order by gross_profit DESC;
              product_name                | gross_profit
------------------------------------------+--------------
 Intel SSDPECME040T401                    |      1744.33
 PNY VCQP6000-PB                          |      1441.00
 PNY VCQM6000-PB                          |       749.95
 Intel Xeon E5-2698 V3 (OEM/Tray)         |       625.54
 AMD 100-505989                           |       571.32
 PNY VCQK6000-PB                          |       550.48
 ATI FirePro S9150                        |       549.38
 Intel Xeon E5-2699 V3 (OEM/Tray)         |       542.95
 Intel Xeon E5-2637 V2 (OEM/Tray)         |       526.88
 PNY VCQM6000-24GB-PB                     |       519.86
 AMD FirePro W9100                        |       515.51
 Intel Xeon E5-2695 V3 (OEM/Tray)         |       506.82
 Intel Xeon E5-2667 V3 (OEM/Tray)         |       505.38
 Intel Xeon E5-2690 V4                    |       495.23
 PNY VCQM5000-PB                          |       491.57
 Intel Xeon E5-2695 V4                    |       489.64
 Intel Xeon E5-2685 V3 (OEM/Tray)         |       489.58
 EVGA 12G-P4-3992-KR                      |       486.92
 Intel Xeon E5-2643 V4 (OEM/Tray)         |       483.27
 Crucial                                  |       481.76
 Intel Xeon E5-2683 V4 (OEM/Tray)         |       475.06
--More--
```

（12）查询雇员表中每个雇员对应的经理名，要求第一列字段名为 employee，第二列字段名为 manager，无其他字段。

思路：employees 表连接到自身。即 INNER JOIN。因为一个表只能在查询中出现一次，所以必须使用表别名为员工提供两个不同的名称，即 e 员工和 m 经理，选择 e 员工的 first_name，别名为 employee，选择 m 员工的 first_name，别名为 manager，当 m 的 employee.id 和 e 的 manager_id 相等时满足雇佣关系，使用 INNER JOIN 连接。

```
SELECT
  e.first_name employee,
  m.first_name manager
FROM employees e
INNER JOIN employees m ON  m.employee_id = e.manager_id;
```



employee	manager
Blake	Tommy
Jude	Tommy
Mohammad	Jude
Harper	Jude
Louie	Blake
Nathan	Louie
Bobby	Louie
Charles	Louie
Gabriel	Louie
Emma	Tommy
Amelie	Emma
Gracie	Jude
Rory	Tommy
Isabella	Tommy
Jessica	Tommy
Ella	Tommy
Ava	Tommy
Mia	Tommy
Evie	Ava
Scarlett	Ella
Ruby	Ella
Chloe	Ella
Isabelle	Ella
Daisy	Ella
Freya	Ella
Elizabeth	Mia
Florence	Jessica
Alice	Jessica
Charlotte	Jessica
Sienna	Jessica
Matilda	Jessica
Evelyn	Isabella
Eva	Isabella
Millie	Isabella
Sofia	Isabella

--More--

（13）查询产品表中所有以 Asus 开头的产品名和价格，并以价格降序显示。SELECT product_name,list_price FROM products WHERE product_name LIKE 'Asus%' ORDER BY list_price DESC;

```
sale=> SELECT product_name,list_price FROM products WHERE product_name LIKE 'Asus%' ORDER BY list_price DESC;
             product_name              | list_price
---------------------------------------+------------
 Asus GTX780TI-3GD5                    |     899.99
 Asus ROG-POSEIDON-GTX1080TI-P11G-GAMING |     864.98
 Asus STRIX-GTX1080TI-O11G-GAMING      |     829.99
 Asus X99-E-10G WS                     |     649.00
 Asus ROG MAXIMUS IX EXTREME           |     573.99
 Asus RAMPAGE V EXTREME                |     572.96
 Asus Z10PE-D8 WS                      |     561.59
 Asus Rampage V Edition 10             |     519.99
 Asus PRIME X299-DELUXE                |     487.30
 Asus X99-E WS/USB 3.1                 |     482.49
 Asus Z10PE-D16 WS                     |     469.99
 Asus X99-DELUXE/U3.1                  |     440.30
 Asus KGPE-D16                         |     417.98
 Asus Z10PE-D16                        |     402.99
 Asus MAXIMUS IX FORMULA               |     388.99
 Asus X99-DELUXE II                    |     383.98
 Asus MAXIMUS VIII EXTREME/ASSEMBLY    |     353.98
 Asus STRIX X299-E GAMING              |     349.99
 Asus TUF X299 MARK 1                  |     339.99
 Asus Z170-WS                          |     338.99
 Asus ROG STRIX X99 GAMING             |     319.99
 Asus SABERTOOTH X99                   |     312.67
 Asus PRIME X299-A                     |     309.85
 Asus MAXIMUS IX CODE                  |     298.98
 Asus Sabertooth 990FX                 |     295.72
 Asus VANGUARD B85                     |     287.00
(26 rows)
```

（14）查询联系表中电话号码不是以‘+1’开头的名、姓和电话号码，并以名升序显示： SELECT first_name,last_name,phone FROM contacts WHERE phone NOT LIKE '+1%' ORDER BY first_name ASC;

```
 first_name | last_name  |      phone
------------+------------+------------------
 Adah       | Myers      | +41 3 012 3553
 Adam       | Jacobs     | +91 80 012 3699
 Adrienne   | Lang       | +39 2 012 4771
 Aleshia    | Reese      | +41 4 012 3563
 Alessandra | Estrada    | +41 56 012 3527
 Amber      | Brady      | +91 80 012 3837
 Annabelle  | Butler     | +91 80 012 3737
 Annelle    | Lawrence   | +39 10 012 4379
 Arlette    | Thornton   | +91 80 012 3719
 Barbie     | Carter     | +41 5 012 3573
 Basilia    | Downs      | +66 76 012 4441
 Beatrice   | Ford       | +91 80 012 4785
 Bill       | Stein      | +39 6 012 4501
 Blanche    | Robbins    | +39 6 012 4389
 Bobby      | Wilson     | +91 11 012 4817
 Brandie    | Buchanan   | +91 22 012 4831
 Carita     | Mcintyre   | +86 10 012 4165
 Carlos     | Moody      | +86 811 012 4093
 Cathey     | Mcdowell   | +41 65 012 3545
 Charlene   | Booker     | +41 61 012 3537
 Charlsie   | Carey      | +91 80 012 3731
 Cleo       | English    | +41 8 012 3575
 Colette    | Estrada    | +81 565 012 4567
 Colleen    | Estrada    | +91 80 012 3741
 Collene    | Newton     | +39 49 012 4373
 Cristine   | Bell       | +91 11 012 4851
 Daina      | Combs      | +91 80 012 3715
 Darron     | Robertson  | +91 80 012 3723
 Deneen     | Hays       | +41 66 012 3547
 Dick       | Lamb       | +91 80 012 4803
 Don        | Hansen     | +39 10 012 4385
 Dorotha    | Wong       | +41 6 012 3565
 Dustin     | Paul       | +39 6 012 4511
 Ed         | Mueller    | +91 80 012 3739
 Elicia     | Townsend   | +41 58 012 3531
--More--
```

（15）查询联系表中的电话号码和电子邮件，要求名中包含‘Je_i’且以名升序显示。

```
sale=> SELECT first_name,phone,email FROM contacts WHERE first_name LIKE 'Je_i' ORDER BY first_name ASC;
 first_name |      phone      |         email
------------+-----------------+----------------------
 Jeni       | +1 812 123 4129 | jeni.levy@centene.com
 Jeri       | +49 90 012 4131 | jeri.randall@nike.com
(2 rows)
```

（16）查询联系表中所有以开头'Je'的名，且至少包含 3 个字符的名，姓，电子邮件和电话：SELECT first_name,last_name,email,phone FROM contacts WHERE first_name LIKE 'Je_%';

```
sale=> SELECT first_name,last_name,email,phone FROM contacts WHERE first_name LIKE 'Je_%';
 first_name | last_name |              email               |      phone
------------+-----------+----------------------------------+-----------------
 Jeni       | Levy      | jeni.levy@centene.com            | +1 812 123 4129
 Jessika    | Merritt   | jessika.merritt@bnymellon.com    | +1 612 123 4397
 Jeri       | Randall   | jeri.randall@nike.com            | +49 90 012 4131
 Jermaine   | Cote      | jermaine.cote@wfscorp.com        | +49 91 012 4133
 Jeannie    | Poole     | jeannie.poole@aboutmcdonalds.com | +91 80 012 4637
 Jess       | Nguyen    | jess.nguyen@searsholdings.com    | +39 2 012 4773
 Jerica     | Brooks    | jerica.brooks@northropgrumman.com | +91 11 012 4811
 Jen        | Mcmahon   | jen.mcmahon@voya.com             | +41 68 012 3571
(8 rows)
```

（17）查询订单表中所有没有销售员负责的订单（i.e., query all sales orders that do not have a responsible salesman）

SELECT *FROM orders WHERE salesman_id is null;

order_id	customer_id	status	salesman_id	order_date
2	4	Shipped		2015-04-26 00:00:00
3	5	Shipped		2017-04-26 00:00:00
6	6	Shipped		2015-04-09 00:00:00
7	7	Shipped		2017-02-15 00:00:00
8	8	Shipped		2017-02-14 00:00:00
9	9	Shipped		2017-02-14 00:00:00
10	44	Pending		2017-01-24 00:00:00
11	45	Shipped		2016-11-29 00:00:00
12	46	Shipped		2016-11-29 00:00:00
13	47	Shipped		2016-11-29 00:00:00
14	48	Shipped		2017-09-28 00:00:00
15	49	Shipped		2017-09-27 00:00:00
16	16	Pending		2016-09-27 00:00:00
17	17	Shipped		2017-09-27 00:00:00
18	18	Shipped		2016-08-16 00:00:00
19	19	Shipped		2016-05-27 00:00:00
20	20	Shipped		2017-05-27 00:00:00
21	21	Pending		2017-05-27 00:00:00
22	22	Canceled		2016-05-26 00:00:00
23	23	Shipped		2016-09-07 00:00:00
24	41	Shipped		2016-09-07 00:00:00
25	42	Shipped		2016-08-24 00:00:00
27	43	Canceled		2016-08-16 00:00:00
29	44	Shipped		2017-08-14 00:00:00
30	45	Shipped		2017-08-12 00:00:00
31	46	Canceled		2017-08-12 00:00:00
32	47	Shipped		2017-03-09 00:00:00
33	48	Shipped		2017-03-07 00:00:00
34	49	Shipped		2016-06-12 00:00:00
35	50	Shipped		2016-09-05 00:00:00
36	51	Shipped		2017-09-05 00:00:00
37	52	Shipped		2017-02-19 00:00:00
68	9	Pending		2016-06-13 00:00:00
73	48	Shipped		2016-02-17 00:00:00
75	16	Shipped		2017-02-10 00:00:00

--More--

（18）统计每个顾客的订单总数（查询订单表）

```
SELECT name, COUNT( order_id ) FROM orders
INNER JOIN customers USING(customer_id)
GROUP BY name ORDER BY name;
```

name	count
AECOM	1
AbbVie	4
Abbott Laboratories	1
Aflac	4
Alcoa	4
American Electric Power	1
AutoNation	4
AutoZone	1
Baker Hughes	1
Bank of New York Mellon Corp.	1
Becton Dickinson	1
Bristol-Myers Squibb	1
Centene	4
CenturyLink	5
Colgate-Palmolive	1
Community Health Systems	4
ConAgra Foods	1
DTE Energy	1
Dollar General	1
Dollar Tree	1
Eli Lilly	1
Emerson Electric	4
Facebook	1
Freeport-McMoRan	1
Gap	1
General Mills	5
Goodyear Tire & Rubber	1
Health Net	1
International Paper	4
Jabil Circuit	5
Micron Technology	1
NGL Energy Partners	1
NextEra Energy	5
Nucor	1
PG&E Corp.	1

--More--

（19）统计每个订单的总价格大于 1000000 的订单号和总价格，并按总价格降序显示结果（查询订单细节表 order_items，总价格=unit_price*quantity）

思路：HAVING 类似于 WHERE，两者都用于过滤结果记录，但 HAVING 用于过滤聚合数据（当使用 GROUP BY 时）。

```
SELECT order_id, SUM(unit_price*quantity) amount FROM orders
INNER JOIN order_items USING(order_id)
GROUP BY ROLLUP(order_id) HAVING (amount>1000000)
ORDER BY amount DESC ;
```

```
sale=> SELECT order_id, SUM(unit_price*quantity) amount FROM orders
sale-> INNER JOIN order_items USING(order_id)
sale-> GROUP BY ROLLUP(order_id) HAVING (amount>1000000)
sale-> ORDER BY amount DESC ;
 order_id |    amount
----------+---------------
          | 52742092.6400
       70 |  1278962.1700
       46 |  1269323.7700
       78 |  1198331.5900
        1 |  1143716.8700
       68 |  1088670.1200
       27 |  1084871.4900
       32 |  1081679.8800
       92 |  1050939.9700
       59 |  1043144.7200
(10 rows)
```

（20）①创建一个折扣表 discounts

```
sale=> CREATE TABLE discounts
sale->  ( product_id NUMBER, --产品号，主码
sale(>  discount_message VARCHAR2( 255 ) NOT NULL, --折扣信息
sale(>  PRIMARY KEY( product_id ) );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "discounts_pkey" for table "discounts"
CREATE TABLE
```

②插入 3 条数据

```
sale=> INSERT INTO discounts(product_id, discount_message) VALUES(1, 'Buy 1 and Get 25% OFF on 2nd ');
INSERT 0 1
sale=> INSERT INTO discounts(product_id, discount_message) VALUES(2, 'Buy 2 and Get 50% OFF on 3rd ');
INSERT 0 1
sale=> INSERT INTO discounts(product_id, discount_message) VALUES(3, 'Buy 3 Get 1 free');
INSERT 0 1
```

③查询折扣表中折扣信息出现“25%”的产品号和折扣信息

思路：ESCAPE 子句可帮助查找包含一个或多个通配符的字符串。

```
SELECT product_id, discount_message FROM discounts
WHERE discount_message LIKE '%25!%%' ESCAPE '!';
```

```
sale=> SELECT product_id, discount_message FROM discounts
sale-> WHERE discount_message LIKE '%25!%%' ESCAPE '!';
 product_id |       discount_message
------------+------------------------------
          1 | Buy 1 and Get 25% OFF on 2nd
(1 row)
```

3. 实验总结

3.1 实验思考

使用 GROUP BY 时 SELECT 关键词后面的列及其次序有何要求? 请上机验证你的看法。

答: ① GROUP BY 子句出现在 FROM 子句之后。如果出现 WHERE case 子句，则 GROUP BY 子句必须放在 WHERE 子句之后。② SELECT 子句中不能书写聚合键之外的列名。GROUP BY 子句必须仅包含聚合或分组列，聚合结果是无序的。

上机验证：图一说明在 SELECT 子句中书写聚合键之外的列名会发生错误。列名 status 并没有包含在 GROUP BY 子句当中。因此，该列名也不能书写在 SELECT 子句之中。不支持这种语法的原因是通过某个聚合键将表分组之后，结果中的一行数据就代表一组。而如果 SELECT 其他列名，可能出现不是一对一的情况。

```
sale=> SET SEARCH_PATH To icebear,public;
SET
sale=> SELECT order_id, status,SUM(unit_price*quantity) amount FROM orders
sale-> INNER JOIN order_items USING(order_id)
sale-> GROUP BY ROLLUP(order_id) HAVING (amount>1000000);
ERROR:  column "orders.status" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: SELECT order_id, status,SUM(unit_price*quantity) amount FROM...
                         ^
```

图二说明在 WHERE 子句中不能使用聚合函数，只有 SELECT 子句和 HAVING 子句（以及 ORDER BY ）中能够使用 COUNT 等聚合函数。

```
sale=> SELECT order_id as wrong, SUM(unit_price*quantity) amount FROM orders
sale-> INNER JOIN order_items USING(order_id) WHERE amount>1000000
sale-> GROUP BY ROLLUP(order_id);
ERROR:  aggregates not allowed in WHERE clause
LINE 1: SELECT order_id as wrong, SUM(unit_price*quantity) amount FR...
                                  ^
```

3.2 对实验的认识

通过实验我对 openGauss 中的一些语句更熟悉了。如 SET SEARCH_PATH TO icebear, public;可以将搜索路径设置为 icebear、public，首先搜索 icebear。如 SELECT * FROM customer_t1;可以用来查询表 customer_t1 的所有数据。gsql -d sale -p 26000 -U yuxiaoping -W yuxiaoping@123 -r 可以用来将新用户连接到数据库。可以使用 gsql -d postgres -p 26000 -r 连接到 postgres。gs_om -t start 可以开启数据库。

3.3 遇到的困难及解决方法

要更改当前会话的默认 Schema，请使用 SET 命令。执行如下命令 SET SEARCH_PATH To icebear,public;将搜索路径设置为 myschema、public，首先搜索 myschema。

```
sale=> SET SEARCH_PATH TO icebear, public;
SET
```

高斯默认有 session 超时时间，若想要 session 一直保持，需要修改配置项：ALTER DATABASE postgres SET session_timeout TO 0;

```
postgres=# ALTER DATABASE postgres SET session_timeout TO 0;
ALTER DATABASE
```