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:	2023/4/7
实验地点:	文宣楼 A402
提交日期:	2023/4/9

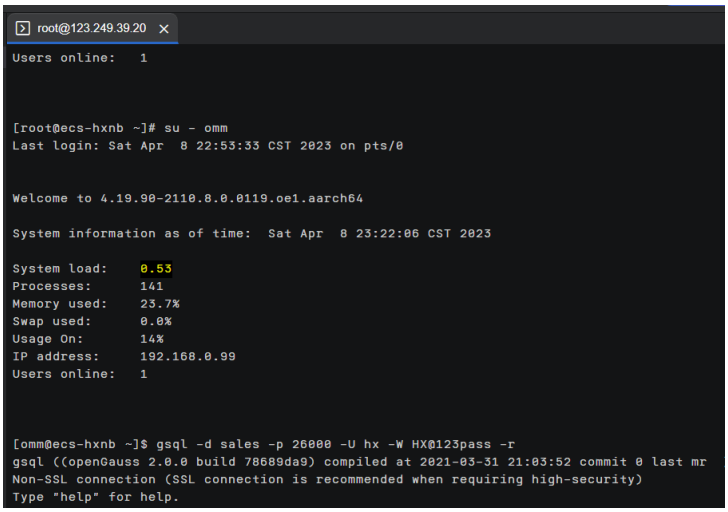
学号:	22920212204392
姓名:	黄勖
专业年级:	软工 2021 级
学年学期:	2022-2023 学年第二学期

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
  - 简单单表查询（此处指不涉及模糊、集合、聚集、分组、排序的查询）
  - 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
- 掌握表别名的使用场合及方法
- 掌握自身连接的使用方法

2.实验内容和步骤

（1）登录 ECS 服务器，以 omm 操作系统管理员身份登录数据库，使用 gsql 连接到数据库。



（2）重建 Sales 数据库，导入上次实验的数据

根据实验 2 的销售数据库 Sales，包含 12 张表，加入各表的表名和信息，约束。

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

select customer_id, name, credit_limit from customers;

```
sales=> select customer_id, name, credit_limit from customers;
 customer_id |                name                | credit_limit
-------------+------------------------------------+--------------
         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                           |      3600.00
         212 | Qualcomm                           |       500.00
         216 | EMC                                |       700.00
         220 | Time Warner Cable                  |      3700.00
         223 | Northrop Grumman                   |      5000.00
          39 | Lear                               |       500.00
          43 | Facebook                           |       500.00
          46 | Supervalu                          |       500.00
          49 | NextEra Energy                     |       600.00
          52 | PG&E Corp.                         |       700.00
          55 | Goodyear Tire & Rubber             |       700.00
```

（4）查询顾客的所有信息，且只显示前 20 条记录。--需要用到 limit n 来限制输出记录数

select * from customers limit 20;

```
sales=> select * from customers limit 20;
 customer_id |            name            |                address                |                website                | credit_limit
-------------+----------------------------+---------------------------------------+---------------------------------------+--------------
         177 | United Continental Holdings | 2904 S Salina St, Syracuse, NY        | http://wwwunitedcontinentalholdingscom |      5000.00
         180 | INTL FCStone               | 5344 Haverford Ave, Philadelphia, PA  | http://wwwintlfcstonecom              |      5000.00
         184 | Publix Super Markets       | 1795 Wu Meng, Muang Chonburi,         | http://wwwpublixcom                   |      1200.00
         187 | ConocoPhillips             | Walpurgisstr 69, Munich,              | http://wwwconocophillipscom           |      2400.00
         190 | 3M                         | Via Frenzy 6903, Roma,                | http://www3mcom                       |      1200.00
         192 | Exelon                     | Via Luminosa 162, Firenze,            | http://wwwexeloncorpcom               |       500.00
         208 | Tesoro                     | Via Notoriosa 1942, Firenze,          | http://wwwtsocorpcom                  |       500.00
         207 | Northwestern Mutual        | 1831 No Wong, Peking,                 | http://wwwnorthwesternmutualcom       |      3600.00
         200 | Enterprise Products Partners | Via Notoriosa 1949, Firenze,        | http://wwwenterpriseproductscom       |      2400.00
         204 | Rite Aid                   | Piazza Cacchiatore 23, San Giminiano, | http://wwwriteaidcom                  |      3600.00
         212 | Qualcomm                   | Piazza Svizzera, Milano,              | http://wwwqualcommcom                 |       500.00
         216 | EMC                        | Via Delle Grazie 11, San Giminiano,   | http://wwwemccom                      |       700.00
         220 | Time Warner Cable          | 1597 Legend St, Mysore, Kar           | http://wwwtwccom                      |      3700.00
         223 | Northrop Grumman           | 1606 Sangam Blvd, New Delhi,          | http://wwwnorthropgrummancom          |      5000.00
          39 | Lear                       | 2115 N Towne Ln Ne, Cedar Rapids, IA  | http://wwwlearcom                     |       500.00
          43 | Facebook                   | 5112 Sw 9Th St, Des Moines, IA        | http://wwwfacebookcom                 |       500.00
          46 | Supervalu                  | 8989 N Port Washington Rd, Milwaukee, WI | http://wwwsupervalucom             |       500.00
          49 | NextEra Energy             | 4715 Sprecher Rd, Madison, WI         | http://wwwnexteraenergycom            |       600.00
          52 | PG&E Corp.                 | 8600 W National Ave, Milwaukee, WI    | http://wwwpgecom                      |       700.00
          55 | Goodyear Tire & Rubber     | 600 N Broadway Fl 1, Milwaukee, WI    | http://wwwgoodyearcom                 |       700.00
```

（5）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 by：排序
ASC：升序
DESC：降序

```
sales=> 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
       57 |          68 | Shipped  | 2017-09-24 00:00:00
       10 |          44 | Pending  | 2017-09-24 00:00:00
       28 |           6 | Canceled | 2017-09-15 00:00:00
       29 |          44 | Shipped  | 2017-09-14 00:00:00
       31 |          46 | Canceled | 2017-09-12 00:00:00
       30 |          45 | Shipped  | 2017-09-12 00:00:00
       99 |          49 | Shipped  | 2017-09-07 00:00:00
       36 |          51 | Shipped  | 2017-09-05 00:00:00
      100 |          16 | Pending  | 2017-09-05 00:00:00
      101 |           3 | Pending  | 2017-09-03 00:00:00
       77 |           1 | Shipped  | 2017-09-02 00:00:00
       60 |           1 | Shipped  | 2017-06-30 00:00:00
       21 |          21 | Pending  | 2017-05-27 00:00:00
       20 |          20 | Shipped  | 2017-05-27 00:00:00
```

（6）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;
```

```
sales=> 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
```

(7) 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;

```
sales=> 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)
```

这个查询将按照 city 和 state 列的值进行升序排序。如果 state 列中有 NULL 值，则这些 NULL 值将在排序后显示在最后。

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

```
sales=> 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)
```

这个查询将按照 state 列的值进行升序排序。如果 state 列中有 NULL 值，则这些 NULL 值将在排序后显示在最前面。

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

```
sales=> 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)
```

这个查询将按照 state 列的值进行升序排序。如果 state 列中有 NULL 值，则这些 NULL 值将在排序后显示在最后面。

（8）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;
```

```
sales=> 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
```

（9）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';

```
sales=> 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 |   74163.00
 Kingston     | Speed:DDR3-1333,Type:240-pin DIMM,CAS:9Module:4x16GBSize:64GB  |   67138.00
 Kingston     | Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x8GBSize:32GB  |    6535.00
 Kingston     | Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x16GBSize:64GB |     644.00
(4 rows)
```

（10）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_id, list_price from products where list_price > 500 AND category_id = 4;

```
sales=> select product_id, list_price from products where list_price > 500 AND category_id = 4;
 product_id | list_price
------------+------------
        190 |   94899.00
         62 |   78979.00
        189 |     649.00
        191 |   57399.00
         63 |   57296.00
        193 |   56159.00
        271 |   54959.00
         64 |   52599.00
        192 |   51999.00
        225 |   50398.00
        252 |   49999.00
        274 |   49999.00
         61 |    4873.00
        188 |   48249.00
         28 |   47999.00
```

（11）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 standard_cost ASC;

```
sales=> select product_name, list_price from products where list_price between 650 and 680 order by standard_cost ASC;
    product_name     | list_price
---------------------+------------
 Intel Core i7-3930K |     660.00
(1 row)
```

（12）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;

```
sales=> SELECT first_name "First_Name", last_name "Family_Name" FROM employees;
 "first_name" | "family_name"
--------------+---------------
 Summer       | Payne
 Rose         | Stephens
 Annabelle    | Dunn
 Tommy        | Bailey
 Blake        | Cooper
 Jude         | Rivera
 Tyler        | Ramirez
 Ryan         | Gray
 Elliot       | Brooks
 Elliott      | James
 Albert       | Watson
 Mohammad     | Peterson
 Harper       | Spencer
 Louie        | Richardson
 Nathan       | Cox
 Bobby        | Torres
```

（13）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 list_price-standard_cost DESC;

```
sales=> select product_name, list_price-standard_cost "gross_profit" FROM products ORDER BY list_price-standard_cost DESC;
                product_name                | "gross_profit"
--------------------------------------------+----------------
 Intel Xeon E5-2697 V4                      |      234055.00
 Intel SSDPECME040T401                      |      174433.00
 PNY VCQP6000-PB                            |      144100.00
 Intel Core i7-5960X (OEM/Tray)             |       89001.00
 PNY VCQP4000-PB                            |       82265.00
 PNY VCQM6000-PB                            |       74995.00
 Corsair Dominator Platinum                 |       73494.00
 GSkill Trident Z                           |       70331.00
 Zotac ZT-P10810D-10P                       |       70198.00
 Gigabyte GV-N108TAORUS-11GD                |       68445.00
 Intel Core i7-980                          |       64362.00
 Intel Core i7-7820X                        |       62764.00
 Intel Xeon E5-2698 V3 (OEM/Tray)           |       62554.00
 GSkill Ripjaws V Series                    |       62075.00
 Corsair Vengeance LPX                      |       58882.00
 AMD 100-505989                             |       57132.00
 Asus RAMPAGE V EXTREME                     |       56837.00
 PNY VCQK6000-PB                            |       55048.00
```

（14）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_name，第二列字段名为 manager_name（雇员和经理的姓名同一格式为‘first_name, last_name’）。

说明：此处查询涉及字符串的连接。字符串连接符号为“||”，即两个连续的竖线。如

```
openGauss=# SELECT 'MPP'||'DB' AS RESULT;
result
--------
 MPPDB
(1 row)
```

注：字符串连接符||的两边只能用单引号，如使用双引号等其它符号将会报错

select first.first_name||','||first.last_name as employee_name, second.first_name||','||second.last_name as manager_name from employees first, employees second WHERE first.manager_id = second.employee_id;

```
sales=> select first.first_name||','||first.last_name as employee_name, second.first_name||','||second.last_name as manager_name from employees first,
 employees second WHERE first.manager_id = second. employee_id;
   employee_name   |   manager_name
-------------------+-------------------
 Summer,Payne      | Rose,Stephens
 Rose,Stephens     | Jude,Rivera
 Annabelle,Dunn    | Jude,Rivera
 Blake,Cooper      | Tommy,Bailey
 Jude,Rivera       | Tommy,Bailey
 Tyler,Ramirez     | Mohammad,Peterson
 Ryan,Gray         | Mohammad,Peterson
 Elliot,Brooks     | Mohammad,Peterson
 Elliott,James     | Mohammad,Peterson
 Albert,Watson     | Mohammad,Peterson
 Mohammad,Peterson | Jude,Rivera
 Harper,Spencer    | Jude,Rivera
 Louie,Richardson  | Blake,Cooper
 Nathan,Cox        | Louie,Richardson
 Bobby,Torres      | Louie,Richardson
 Charles,Ward      | Louie,Richardson
 Gabriel,Howard    | Louie,Richardson
 Emma,Perkins      | Tommy,Bailey
 Amelie,Hudson     | Emma,Perkins
```

（15）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;

模糊查询：like， rlike

%：任意字符串

_：规定大小，一个横线表示一个字符

```
sales=> select product_name,list_price from products where product_name like 'Asus%' order by list_price desc;
             product_name              | list_price
---------------------------------------+------------
 Asus GTX780TI-3GD5                    |   89999.00
 Asus ROG-POSEIDON-GTX1080TI-P11G-GAMING |   86498.00
 Asus STRIX-GTX1080TI-O11G-GAMING      |   82999.00
 Asus ROG MAXIMUS IX EXTREME           |   57399.00
 Asus RAMPAGE V EXTREME                |   57296.00
 Asus Z10PE-D8 WS                      |   56159.00
 Asus Rampage V Edition 10             |   51999.00
 Asus X99-E WS/USB 31                  |   48249.00
 Asus Z10PE-D16 WS                     |   46999.00
 Asus KGPE-D16                         |   41798.00
 Asus Z10PE-D16                        |   40299.00
 Asus MAXIMUS IX FORMULA               |   38899.00
 Asus X99-DELUXE II                    |   38398.00
 Asus MAXIMUS VIII EXTREME/ASSEMBLY    |   35398.00
 Asus STRIX X299-E GAMING              |   34999.00
 Asus TUF X299 MARK 1                  |   33999.00
 Asus Z170-WS                          |   33899.00
 Asus ROG STRIX X99 GAMING             |   31999.00
 Asus SABERTOOTH X99                   |   31267.00
 Asus PRIME X299-A                     |   30985.00
 Asus MAXIMUS IX CODE                  |   29898.00
 Asus Sabertooth 990FX                 |   29572.00
 Asus PRIME X299-DELUXE                |    4873.00
 Asus X99-DELUXE/U31                   |    4403.00
 Asus X99-E-10G WS                     |     649.00
 Asus VANGUARD B85                     |     287.00
(26 rows)
```

（16）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 last_name;

```
sales=> select first_name, last_name, phone from contacts where phone not like '+1%' order by last_name;
 first_name | last_name  |      phone
------------+------------+-----------------
 Lucius     | Abbott     | +91 80 012 4783
 Jarvis     | Allison    | +39 10 012 4371
 Ginger     | Atkinson   | +91 33 012 4825
 Marhta     | Baldwin    | +39 49 012 4409
 Yolanda    | Ball       | +91 80 012 4809
 Maple      | Barnett    | +91 80 012 3725
 Fran       | Battle     | +91 80 012 4123
 Shamika    | Bauer      | +91 11 012 4853
 Josiah     | Beasley    | +91 80 012 3735
 Cristine   | Bell       | +91 11 012 4851
 Shenna     | Blair      | +91 141 012 4895
 Stephaine  | Booker     | +39 55 012 4559
 Charlene   | Booker     | +41 61 012 3537
 Jaleesa    | Bowen      | +66 76 012 4633
 Amber      | Brady      | +91 80 012 3837
 Shelia     | Brewer     | +49 89 012 4129
 Jerica     | Brooks     | +91 11 012 4811
 Marianne   | Bryant     | +39 6 012 4507
 Brandie    | Buchanan   | +91 22 012 4831
```

（17）查询联系表中的电话号码和电子邮件，要求名(first_name) 的长度为 4 且以'Je'开头，以'i'结尾，按名升序显示。

select phone,email from contacts where first_name like 'Je_i' order by first_name;

```
sales=> select phone,email from contacts where first_name like 'Je_i' order by first_name;
      phone      |        email
-----------------+---------------------
 +1 812 123 4129 | jenilevy@centenecom
 +49 90 012 4131 | jerirandall@nikecom
(2 rows)
```

（18）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_%' ;

```
sales=> select first_name, last_name,email,phone from contacts where first_name like 'Je_%' ;
 first_name | last_name |              email               |      phone
------------+-----------+----------------------------------+-----------------
 Jeni       | Levy      | jenilevy@centenecom              | +1 812 123 4129
 Jessika    | Merritt   | jessikamerritt@bnymelloncom      | +1 612 123 4397
 Jeri       | Randall   | jerirandall@nikecom              | +49 90 012 4131
 Jermaine   | Cote      | jermainecote@wfscorpcom          | +49 91 012 4133
 Jeannie    | Poole     | jeanniepoole@aboutmcdonaldscom   | +91 80 012 4637
 Jess       | Nguyen    | jessnguyen@searsholdingscom      | +39 2 012 4773
 Jerica     | Brooks    | jericabrooks@northropgrummancom  | +91 11 012 4811
 Jen        | Mcmahon   | jenmcmahon@voyacom               | +41 68 012 3571
(8 rows)
```

（19）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;
```

```
sales=> select * from orders where salesman_id IS NULL;
 order_id | customer_id |  status  | salesman_id |     order_date
----------+-------------+----------+-------------+---------------------
        2 |           4 | Shipped  |             | 2015-02-26 00:00:00
        3 |           5 | Shipped  |             | 2017-02-26 00:00:00
        6 |           6 | Shipped  |             | 2015-02-09 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  |             | 2017-09-24 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  |             | 2016-09-16 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
```

（20）统计每个顾客的订单总数（查询订单表）。

```
select customer_id,count(*) from orders group by customer_id;
```

聚合函数：count(), max(), min(), sum(), avg(), round()

```
sales=> select customer_id,count(*) from orders group by customer_id;
 customer_id | count
-------------+-------
          47 |     5
          42 |     1
          57 |     1
           3 |     4
          45 |     5
          23 |     1
           7 |     4
          52 |     1
          64 |     1
           2 |     4
          49 |     5
          66 |     1
          70 |     1
          68 |     1
          16 |     4
          43 |     1
          59 |     1
           8 |     4
```

（21）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）

select order_id, unit_price*quantity as sum_price from order_items where unit_price*quantity > 1000000 order by unit_price*quantity desc;

```
sales=> select order_id, unit_price*quantity as sum_price from order_items where unit_price*quantity > 1000000 order by unit_price*quantity desc;
 order_id |   sum_price
----------+--------------
       70 | 60302332.0000
       32 | 38985570.0000
       46 | 36858384.0000
       54 | 36719864.0000
       92 | 35441967.0000
       75 | 34487235.0000
       59 | 32803842.0000
       18 | 31865859.0000
      104 | 31552861.0000
       67 | 31187862.0000
       90 | 30904112.0000
       40 | 30768351.0000
       46 | 30239888.0000
       32 | 30027543.0000
       43 | 30013918.0000
       94 | 29969889.0000
```

（22）创建一个折扣表 discounts

```
CREATE TABLE discounts
    (product_id NUMBER,
    discount_message VARCHAR2( 255 ) NOT NULL,
```

PRIMARY KEY(product_id));

```
sales=> CREATE TABLE discounts
sales-> (product_id NUMBER,
sales(> discount_message VARCHAR2( 255 ) NOT NULL,
sales(> PRIMARY KEY( product_id ) );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "discounts_pkey" for table "discounts"
CREATE TABLE
```

插入 3 条数据：

INSERT INTO discounts(product_id, discount_message) VALUES(1, 'Buy 1 and Get 25% OFF on 2nd ');

INSERT INTO discounts(product_id, discount_message) VALUES(2, 'Buy 2 and Get 50% OFF on 3rd ');

INSERT INTO discounts(product_id, discount_message) VALUES(3, 'Buy 3 Get 1 free');

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

要求：查询折扣表中折扣信息出现“25%”的产品号和折扣信息。

select product_id,discount_message from discounts where discount_message like '%25\%%' escape'\';

```
sales=> select product_id,discount_message from discounts 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 完成的工作

设计正确的 SQL 查询语句并测试其结果是否满足查询要求。

3.2 对实验的认识

(1)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。

在使用 GROUP BY 时，SELECT 关键词后面的列必须是 GROUP BY 后面的分组列或聚合函数，否则会出现语法错误。此外，列的次序可以任意排列。

同时，要注意分组列可以是任何类型的表达式，但不能是聚集函数或者子查询。

(2) 单引号和双引号

单引号用来标识实际的值，双引号用来标识表名或列名等数据库中存在的值。当在程序中进行 sql 拼装的时候，加上引号的好处在于可以简化对值的校验，同时又可以避免 sql 注入，提高了安全性。

(3) 收获

这一次实验反复使用数据库查询语言，在加深理解的同时，我提高了运用数据库代码实现自己相关需求的能力，未来在组织数据库查询语言方面我会更加熟练。

对于具体的使用方法，我学会了 SELECT 语句查询表中的数据，使用 WHERE 子句指定查询条件、使用 ORDER BY 子句对查询结

果进行排序、使用 DISTINCT 关键字确保查询结果中不包含重复的元组、使用 AS 关键字将列名重命名为所需的名称、使用 LIMIT 子句限制输出记录数、使用 BETWEEN 关键字指定范围、以及如何使用数学运算符计算列之间的差值，除此之外我还学会了字符串连接符||的使用。

3.3 遇到的困难及解决方法

1. is null 与=null

   SQL 判断字是否为空的正确方法是使用 is NULL 和 is not NULL

   类似 NULL=NULL 和 NULL<>NULL 的用法都是错误的

2. 关于字体

   华文仿宋的字体单引号双引号不能被数据库识别！