数据库系统课程实验报告

实验名称:	实验四 – 数据高级查询
实验日期:	2023/4/21
实验地点:	文宣楼 A402
提交日期:	2023/4/21

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

- 熟练掌握设计正确的 SQL 查询语句以实现数据高级查询的方法
- 熟练掌握 openGauss 连接查询、子查询和集合查询的语法结构及使用方法
  - （内）连接、（全）外连接、左外连接、右外连接
  - 子查询（嵌套查询）
  - 不相关子查询与相关子查询
  - EXISTS/NOT EXISTS
  - ANY
  - ALL
  - 集合运算：UNION、INSERSECT、MINUS/EXCEPT
- 理解不相关子查询与相关子查询的不同，掌握构造相应 SQL 语句的方法
- 熟练掌握基于派生表的查询方法
- 建议：对同一查询要求尽量使用不同的查询语句实现。如，所有带 IN 谓词、比较运算符、ANY 或 ALL 谓词的子查询都能用带 EXISTS 谓词的子查询等价替换。

2.实验内容和步骤

（0）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。

su - omm

gs_om -t start

gsql -d sales -p 26000 -U hx -W HX@123pass -r

```
root@123.249.39.20
Welcome to Huawei Cloud Service
Last login: Sat Apr 22 01:47:20 2023 from 121.36.59.153


Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time:  Sat Apr 22 01:48:55 CST 2023

System load:    0.31
Processes:      146
Memory used:    23.2%
Swap used:      0.0%
Usage On:       14%
IP address:     192.168.0.99
Users online:   1



[root@ecs-hxnb ~]# su - omm
Last login: Sat Apr 22 01:47:25 CST 2023 on pts/0


Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time:  Sat Apr 22 01:48:59 CST 2023

System load:    0.36
Processes:      149
Memory used:    23.5%
Swap used:      0.0%
Usage On:       14%
IP address:     192.168.0.99
Users online:   1



[omm@ecs-hxnb ~]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] ecs-hxnb:
[2023-04-22 01:49:01.460][5864][][gs_ctl]: gs_ctl started,datadir is /gaussdb/data/db1
[2023-04-22 01:49:01.465][5864][][gs_ctl]:  another server might be running; Please use the restart command
=========================================
Successfully started.
[omm@ecs-hxnb ~]$ gsql -d sales -p 26000 -U hx -W HX@123pass -r
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 21:03:52 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.
```

SET search_path TO sales;

```
sales=> SET search_path TO sales;
SET
```

（1）创建两张表 palette_a 和 palette_b（结构相同，但表名不同，color 为颜色）

CREATE TABLE palette_a

(id INT PRIMARY KEY,

color VARCHAR2 (100) NOT NULL);

CREATE TABLE palette_b

(id INT PRIMARY KEY,

color VARCHAR2 (100) NOT NULL);

```
sales=> CREATE TABLE palette_a
sales-> (id INT PRIMARY KEY,
sales(> color VARCHAR2 (100) NOT NULL);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "palette_a_pkey" for table "palette_a"
CREATE TABLE
sales=> CREATE TABLE palette_b
sales-> (id INT PRIMARY KEY,
sales(> color VARCHAR2 (100) NOT NULL);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "palette_b_pkey" for table "palette_b"
CREATE TABLE
```

（2）为表 palette_a 添加样例数据：{(1, 'Red'), (2, 'Green'), (3, 'Blue'), (4, 'Purple')}。

INSERT INTO sales.palette_a(id,color) VALUES(1, 'Red'), (2, 'Green'), (3, 'Blue'), (4, 'Purple');

```
sales=> INSERT INTO sales.palette_a(id,color) VALUES(1, 'Red'), (2, 'Green'), (3, 'Blue'), (4, 'Purple');
INSERT 0 4
```

（3）为表 palette_b 添加样例数据：{(1, 'Green'), (2, 'Red'), (3, 'Cyan'), (4, 'Brown')}。

INSERT INTO sales.palette_b(id,color) VALUES(1, 'Green'), (2,'Red'), (3, 'Cyan'), (4,'Brown');

```
sales=> INSERT INTO sales.palette_b(id,color) VALUES(1, 'Green'), (2,'Red'), (3, 'Cyan'), (4,'Brown' );
INSERT 0 4
```

（4）查询两张表中相同颜色的所有信息。

SELECT * FROM sales.palette_b INNER JOIN sales.palette_a ON sales.palette_a.color=sales.palette_b.color;

```
sales=> SELECT * FROM sales.palette_b INNER JOIN sales.palette_a ON sales.palette_a.color=sales.palette_b.color;
 id | color | id | color
----+-------+----+-------
  1 | Green |  2 | Green
  2 | Red   |  1 | Red
(2 rows)
```

（5）查询 palette_a 表中颜色不出现在 palette_b 表中的 id 和颜色。

实现要求：左外连接（必须）+其它查询方法（如果找到）

SELECT sales.palette_a.* FROM sales.palette_a LEFT OUTER JOIN sales.palette_b ON sales.palette_a.color = sales.palette_b.color where sales.palette_b.color IS NULL;

```
sales=> SELECT sales.palette_a.* FROM sales.palette_a LEFT OUTER JOIN sales.palette_b ON sales.palette_a.color = sal
es.palette_b.color where sales. palette_b.color IS NULL;
 id | color
----+--------
  3 | Blue
  4 | Purple
(2 rows)
```

（6）查询 palette_b 表中颜色不出现在 palette_a 表中的 id 和颜色。

实现要求：右外连接（必须）+其它查询方法（如果找到）

SELECT sales.palette_b.* FROM sales.palette_a RIGHT OUTER JOIN sales.palette_b ON sales.palette_a.color = sales.palette_b.color where sales.palette_a.color IS NULL;

```
sales=> SELECT sales.palette_b.* FROM sales.palette_a RIGHT OUTER JOIN sales.palette_b ON sales.palette_a.color = sa
les.palette_b.color where sales. palette_a.color IS NULL;
 id | color
----+-------
  3 | Cyan
  4 | Brown
(2 rows)
```

（7）查询（5）或（6）两种情况的信息（用（全）外连接）。

SELECT * FROM sales.palette_a FULL OUTER JOIN sales.palette_b ON sales.palette_a.color = sales.palette_b.color WHERE sales.palette_a.color IS NULL or sales.palette_b.color IS NULL;

```
sales=> SELECT * FROM sales.palette_a FULL OUTER JOIN sales.palette_b ON sales.palette_a.color = sales.palette_b.col
or WHERE sales.palette_a.color IS NULL or sales.palette_b.color IS NULL;
 id | color  | id | color
----+--------+----+-------
  3 | Blue   |    |
  4 | Purple |    |
    |        |  3 | Cyan
    |        |  4 | Brown
(4 rows)
```

- 子查询（Subquery）

（8）查询产品表 products 中的 product_id, product_name, list_price 信息，要求产品定价 list_price 大于其平均定价 list_price。

SELECT product_id, product_name, list_price from sales.products where list_price>(select avg(list_price) from sales.products);

```
sales=> SELECT product_id, product_name, list_price from sales.products where list_price>(select avg(list_price) fro
m sales.products);
 product_id |              product_name              | list_price
------------+----------------------------------------+------------
        228 | Intel Xeon E5-2699 V3 (OEM/Tray)       |  341046.00
        248 | Intel Xeon E5-2697 V3                  |  277498.00
        249 | Intel Xeon E5-2698 V3 (OEM/Tray)       |  266072.00
          2 | Intel Xeon E5-2697 V4                  |  255499.00
         45 | Intel Xeon E5-2685 V3 (OEM/Tray)       |  250169.00
         46 | Intel Xeon E5-2695 V3 (OEM/Tray)       |  243195.00
         47 | Intel Xeon E5-2697 V2                  |  237709.00
         51 | Intel Xeon E5-2695 V4                  |  226999.00
         91 | Intel Xeon E5-2695 V2                  |  225999.00
         93 | Intel Xeon E5-2690 (OEM/Tray)          |  211672.00
         98 | Intel Xeon E5-2687W V3                 |  206499.00
        102 | Intel Xeon E5-2687W V4                 |  204269.00
        158 | Intel Xeon E5-2667 V3 (OEM/Tray)       |  200946.00
        159 | Intel Xeon E5-2690 V4                  |  199449.00
        160 | Intel Xeon E5-2690 V3                  |  190873.00
        163 | Intel Xeon E5-2683 V4                  |  189999.00
        169 | Intel Xeon E5-2683 V4 (OEM/Tray)       |  184489.00
        240 | Intel Core i7-4960X Extreme Edition    |  180597.00
        242 | Intel Xeon E5-1680 V3 (OEM/Tray)       |  175199.00
        243 | Intel Xeon E5-2643 V4 (OEM/Tray)       |  170886.00
         19 | Intel Core i7-6950X (OEM/Tray)         |  170437.00
         52 | Intel Xeon E5-2670 V3                  |  167698.00
        165 | Intel Xeon E5-2680                     |  166661.00
        212 | Intel Xeon E5-2680 V4                  |  163999.00
        166 | Intel Xeon E5-2680 V3 (OEM/Tray)       |  163889.00
         82 | Intel Core i7-6950X                    |  149989.00
        213 | Intel Xeon E5-2643 V3 (OEM/Tray)       |  146996.00
        218 | Intel Xeon E5-2660 V4                  |  138889.00
        219 | Intel Xeon E5-2660 V3                  |  129973.00
         85 | Intel Xeon E5-2660 V3 (OEM/Tray)       |  127499.00
        154 | Intel Xeon E5-2650 V3                  |  120498.00
        209 | Intel Core i7-990X Extreme Edition     |  119999.00
         81 | Intel Xeon E5-2650 V4                  |  109999.00
        211 | Intel Xeon E5-2650                     |  106499.00
```

（9）查询产品表 products 中最便宜产品的 product_id, product_name, list_price。

SELECT product_id, product_name, list_price from sales.products where list_price=(select min(list_price) from sales.products);

```
sales=> SELECT product_id, product_name, list_price from sales.products where list_price=(select min(list_price) fro
m sales.products);
 product_id |   product_name    | list_price
------------+-------------------+------------
        286 | Samsung MZ-V6E500 |     234.00
(1 row)
```

（10）查询没有一个订单的顾客姓名。

实现要求：NOT IN（必须）+其它查询方法（如果找到）

SELECT name from sales.customers where customer_id not in (select customer_id from sales.orders);

```
sales=> SELECT name from sales.customers where customer_id not in (select customer_id from sales.orders);
               name
-----------------------------------
 United Continental Holdings
 INTL FCStone
 Publix Super Markets
 ConocoPhillips
 3M
 Exelon
 Tesoro
 Northwestern Mutual
 Enterprise Products Partners
 Rite Aid
 Qualcomm
 EMC
 Time Warner Cable
 Northrop Grumman
 Lear
 Genuine Parts
 Omnicom Group
 Monsanto
 National Oilwell Varco
 Marriott International
 Kinder Morgan
 Molina Healthcare
 Lincoln National
 CH Robinson Worldwide
 Synnex
 HollyFrontier
 PBF Energy
 Waste Management
 Parker-Hannifin
 Farmers Insurance Exchange
 VF
```

- 相关子查询（correlated subquery）

（11） 查询产品表 products 中产品的 product_id, product_name, list_price，要求产品定价 list_price 大于其同类产品（可由 category_id 表达）的平均定价。

实现要求：相关子查询（必须）+基于派生表的查询（如果找到）

SELECT product_id, product_name, list_price FROM sales.products p1 WHERE list_price>(SELECT AVG(list_price) FROM sales.products p2 WHERE p1. category_id=p2.category_id);

```
sales=> SELECT product_id, product_name, list_price FROM sales.products p1 WHERE list_price>(select avg(list_price)
FROM sales.products p2 WHERE p1. category_id=p2.category_id);
 product_id |                product_name                | list_price
------------+--------------------------------------------+------------
        228 | Intel Xeon E5-2699 V3 (OEM/Tray)           |  341046.00
        248 | Intel Xeon E5-2697 V3                      |  277498.00
        249 | Intel Xeon E5-2698 V3 (OEM/Tray)           |  266072.00
          2 | Intel Xeon E5-2697 V4                      |  255499.00
         45 | Intel Xeon E5-2685 V3 (OEM/Tray)           |  250169.00
         46 | Intel Xeon E5-2695 V3 (OEM/Tray)           |  243195.00
         47 | Intel Xeon E5-2697 V2                      |  237789.00
         51 | Intel Xeon E5-2695 V4                      |  226999.00
         91 | Intel Xeon E5-2695 V2                      |  225999.00
         93 | Intel Xeon E5-2690 (OEM/Tray)              |  211672.00
         98 | Intel Xeon E5-2687W V3                     |  206499.00
        102 | Intel Xeon E5-2687W V4                     |  204269.00
        158 | Intel Xeon E5-2667 V3 (OEM/Tray)           |  200946.00
        159 | Intel Xeon E5-2690 V4                      |  199449.00
        160 | Intel Xeon E5-2690 V3                      |  190873.00
        163 | Intel Xeon E5-2683 V4                      |  189999.00
        169 | Intel Xeon E5-2683 V4 (OEM/Tray)           |  184489.00
        240 | Intel Core i7-4960X Extreme Edition        |  180597.00
        242 | Intel Xeon E5-1680 V3 (OEM/Tray)           |  175199.00
        243 | Intel Xeon E5-2643 V4 (OEM/Tray)           |  170886.00
         19 | Intel Core i7-6950X (OEM/Tray)             |  170437.00
         52 | Intel Xeon E5-2670 V3                      |  167698.00
        165 | Intel Xeon E5-2680                         |  166661.00
        212 | Intel Xeon E5-2680 V4                      |  163999.00
        166 | Intel Xeon E5-2680 V3 (OEM/Tray)           |  163889.00
```

- EXISTS 的使用

（12）查询有订单 order 的所有顾客 customer 姓名（查询涉及 customers 表和 orders 表）。

实现要求：使用 EXISTS（必须）+其它查询方法（如果找到）

SELECT name FROM sales.customers WHERE EXISTS(SELECT customer_id FROM sales.orders WHERE sales.orders.customer_id= sales.customers.customer_id);

```
sales=> SELECT name FROM sales.customers WHERE EXISTS(SELECT customer_id FROM sales.orders WHERE sales.orders.custom
er_id= sales.customers.customer_id);
             name
------------------------------
 Facebook
 Supervalu
 NextEra Energy
 PG&E Corp.
 Goodyear Tire & Rubber
 Micron Technology
 ConAgra Foods
 Bank of New York Mellon Corp.
 Sempra Energy
 Raytheon
 Plains GP Holdings
 US Foods Holding
 AbbVie
 Centene
 Community Health Systems
 Alcoa
 International Paper
 Emerson Electric
 Aflac
 AutoNation
 Progressive
 Abbott Laboratories
 Dollar General
 Tenet Healthcare
 Eli Lilly
 AECOM
 Jabil Circuit
 CenturyLink
 General Mills
 Southern
 Thermo Fisher Scientific
```

• EXISTS 与 IN 的不同

（13）执行以下三条与 NULL 相关的语句，观察各自执行的结果，能否从中得出某些初步结论?

[1] SELECT * FROM customers WHERE customer_id IN (NULL);

```
sales=> SELECT * FROM customers WHERE customer_id IN (NULL);
 customer_id | name | address | website | credit_limit
-------------+------+---------+---------+--------------
(0 rows)
```

结果为空集，无记录返回。

[2] SELECT NULL FROM customers;

```
sales=> SELECT NULL FROM customers;
 ?column?
----------
```

返回所有记录的 NULL 值。每个记录都只包含一个 NULL 值列

[3] SELECT * FROM customers WHERE EXISTS (SELECT NULL FROM customers);

```
sales=> SELECT * FROM customers WHERE EXISTS (SELECT NULL FROM customers);
 customer_id |               name               |                    address                    |
  website                | credit_limit
-------------+----------------------------------+-----------------------------------------------+-------------
-------------------------+--------------
         177 | United Continental Holdings      | 2904 S Salina St, Syracuse, NY                | http://wwwuni
tedcontinentalholdingscom |      5000.00
         180 | INTL FCStone                     | 5344 Haverford Ave, Philadelphia, PA          | http://wwwint
lfcstonecom               |      5000.00
         184 | Publix Super Markets             | 1795 Wu Meng, Muang Chonburi,                 | http://wwwpub
lixcom                    |      1200.00
         187 | ConocoPhillips                   | Walpurgisstr 69, Munich,                      | http://wwwcon
ocophillipscom            |      2400.00
         190 | 3M                               | Via Frenzy 6903, Roma,                        | http://www3mc
om                        |      1200.00
         192 | Exelon                           | Via Luminosa 162, Firenze,                    | http://wwwexe
loncorpcom                |       500.00
         208 | Tesoro                           | Via Notoriosa 1942, Firenze,                  | http://wwwtso
corpcom                   |       500.00
         207 | Northwestern Mutual              | 1831 No Wong, Peking,                         | http://wwwnor
thwesternmutualcom        |      3600.00
         200 | Enterprise Products Partners     | Via Notoriosa 1949, Firenze,                  | http://wwwent
erpriseproductscom        |      2400.00
         204 | Rite Aid                         | Piazza Cacchiatore 23, San Giminiano,         | http://wwwrit
eaidcom                   |      3600.00
         212 | Qualcomm                         | Piazza Svizzera, Milano,                      | http://wwwqua
lcommcom                  |       500.00
         216 | EMC                              | Via Delle Grazie 11, San Giminiano,           | http://wwwemc
com                       |       700.00
         220 | Time Warner Cable                | 1597 Legend St, Mysore, Kar                   | http://wwwtwc
com                       |      3700.00
         223 | Northrop Grumman                 | 1606 Sangam Blvd, New Delhi,                  | http://wwwnor
thropgrummancom           |      5000.00
          39 | Lear                             | 2115 N Towne Ln Ne, Cedar Rapids, IA          | http://wwwlea
rcom                      |       500.00
```

返回整个 customers 表中的所有记录，因为在 EXISTS 子查询中使用 NULL，表示只要子查询返回任何记录，WHERE 子句就会匹配。由于 SELECT NULL FROM customers 在任何情况下都会返回一行记录，所以条件始终为 TRUE，同时筛选条件不做限制，因此返回所有记录。

结论：①在 IN 子句中加入 NULL 将返回空集，因为 NULL 值无法与其他值进行匹配。

②在 SELECT 语句中使用 NULL 将返回含有 NULL 值的记录。

③在 EXISTS 子查询中使用 NULL 将返回所有记录，因为该条件始终为 TRUE。

- NOT EXISTS 的使用

（14）找出所有没有订单的顾客姓名（查询涉及 customers 表和 orders 表）。

实现要求：使用 NOT EXISTS（必须）+其它查询方法（如果找到）

SELECT name FROM sales.customers WHERE NOT EXISTS (SELECT * FROM sales.orders WHERE sales.customers.customer_id= sales.orders.customer_id);

```
sales=> SELECT name FROM sales.customers WHERE NOT EXISTS (SELECT * FROM sales.orders WHERE sales.customers.customer
_id= sales.orders.customer_id);
                name
-----------------------------------
 United Continental Holdings
 INTL FCStone
 Publix Super Markets
 ConocoPhillips
 3M
 Exelon
 Tesoro
 Northwestern Mutual
 Enterprise Products Partners
 Rite Aid
 Qualcomm
 EMC
 Time Warner Cable
 Northrop Grumman
 Lear
 Genuine Parts
 Omnicom Group
 Monsanto
 National Oilwell Varco
 Marriott International
 Kinder Morgan
 Molina Healthcare
 Lincoln National
 CH Robinson Worldwide
 Synnex
 HollyFrontier
 PBF Energy
 Waste Management
 Parker-Hannifin
```

- ANY 的使用

（15）查询产品表 products 中的产品名 product_name 和定价 list_price，要求其定价高于产品种类 1 中的任何产品定价。

实现要求：ANY（必须）+其它查询方法（如果找到）

SELECT product_name, list_price FROM sales.products WHERE list_price>ANY(SELECT list_price FROM sales.products p WHERE p.category_id=1);

```
sales=> SELECT product_name, list_price FROM sales.products WHERE list_price>ANY(SELECT list_price FROM sales.produc
ts p WHERE p.category_id=1);
               product_name                | list_price
-------------------------------------------+------------
 Intel Xeon E5-2699 V3 (OEM/Tray)          |  341046.00
 Intel Xeon E5-2697 V3                     |  277498.00
 Intel Xeon E5-2698 V3 (OEM/Tray)          |  266072.00
 Intel Xeon E5-2697 V4                     |  255499.00
 Intel Xeon E5-2685 V3 (OEM/Tray)          |  250169.00
 Intel Xeon E5-2695 V3 (OEM/Tray)          |  243195.00
 Intel Xeon E5-2697 V2                     |  237709.00
 Intel Xeon E5-2695 V4                     |  226999.00
 Intel Xeon E5-2695 V2                     |  225999.00
 Intel Xeon E5-2643 V2 (OEM/Tray)          |    2200.00
 Intel Xeon E5-2690 (OEM/Tray)             |  211672.00
 Intel Xeon E5-2687W V3                    |  206499.00
 Intel Xeon E5-2687W V4                    |  204269.00
 Intel Xeon E5-2667 V3 (OEM/Tray)          |  200946.00
 Intel Xeon E5-2690 V4                     |  199449.00
 Intel Xeon E5-2690 V3                     |  190873.00
 Intel Xeon E5-2470V2                      |   19047.00
 Intel Xeon E5-2683 V4                     |  189999.00
 Intel Xeon E5-2637 V2 (OEM/Tray)          |    1850.00
 Intel Xeon E5-2683 V4 (OEM/Tray)          |  184489.00
 Intel Core i7-4960X Extreme Edition       |  180597.00
 Intel Xeon E5-2699 V4 (OEM/Tray)          |    1756.00
 Intel Xeon E5-1680 V3 (OEM/Tray)          |  175199.00
 Intel Xeon E5-2643 V4 (OEM/Tray)          |  170886.00
 Intel Core i7-6950X (OEM/Tray)            |  170437.00
 Intel Xeon E5-2670 V3                     |  167698.00
```

- ALL 的使用

（16）查询产品表 products 中的产品名 product_name 和定价 list_price，要求其定价高于产品种类 1 中的所有定价。

SELECT product_name, list_price FROM sales.products WHERE list_price>ALL(SELECT list_price FROM sales.products p WHERE p.category_id=1);

```
sales=> SELECT product_name, list_price FROM sales.products WHERE list_price>ALL(SELECT list_price FROM sales.produc
ts p WHERE p.category_id=1);
     product_name      | list_price
-----------------------+------------
 PNY VCQP6000-PB       |  549999.00
 Intel SSDPECME040T401 |  886799.00
(2 rows)
```

（17）查询产品表 products 中的产品名 product_name 和定价 list_price，要求其定价低于产品种类的所有平均定价。

实现要求：ALL（必须）+其它查询方法（如果找到）

SELECT product_name, list_price FROM sales.products WHERE list_price<ALL(SELECT avg(list_price) FROM sales.products GROUP BY category_id);

```
sales=> SELECT product_name, list_price FROM sales.products WHERE list_price<ALL(SELECT avg(list_price) FROM sales.p
roducts GROUP BY category_id);
           product_name            | list_price
-----------------------------------+------------
 Intel Xeon E5-2643 V2 (OEM/Tray)  |    2200.00
 Intel Xeon E5-2470V2              |   19047.00
 Intel Xeon E5-2637 V2 (OEM/Tray)  |    1850.00
 Intel Xeon E5-2699 V4 (OEM/Tray)  |    1756.00
 Intel Xeon E5-2650 V2             |    1249.00
 Intel Core 2 Extreme QX9775       |     892.00
 Intel Core i7-4770K               |     799.00
 Intel Core i7-3930K               |     660.00
 PNY VCQM6000-24GB-PB              |    4139.00
 ATI FirePro S9050                 |    1699.00
 AMD FirePro S7000                 |   12185.00
 NVIDIA VCQM4000-PB                |     790.00
 Crucial                           |     799.00
 Kingston                          |    6535.00
 GSkill Ripjaws V Series           |    6452.00
 Kingston                          |     644.00
 Asus X99-E-10G WS                 |     649.00
 Asus PRIME X299-DELUXE            |    4873.00
 Asus X99-DELUXE/U31               |    4403.00
 ASRock EP2C612 WS                 |   35849.00
 ASRock Z270 SuperCarrier          |   35398.00
 Asus MAXIMUS VIII EXTREME/ASSEMBLY |   35398.00
 Asus STRIX X299-E GAMING          |   34999.00
 Gigabyte X299 AORUS Ultra Gaming  |   34399.00
 Asus TUF X299 MARK 1              |   33999.00
 Asus Z170-WS                      |   33899.00
 MSI X299 GAMING PRO CARBON AC     |   33781.00
 MSI X99A XPOWER GAMING TITANIUM   |   32999.00
```

- UNION 的使用

（18）查询 contacts 表和 employees 表中的所有 last_name，并以 last_name 升序显示。

实现要求：去重+UNION（必须）+其它查询方法（如果找到）

SELECT last_name FROM sales.contacts UNION SELECT last_name FROM sales. employees ORDER BY last_name ASC;

```
sales=> SELECT last_name FROM sales.contacts UNION SELECT last_name FROM sales. employees ORDER BY last_name ASC;
 last_name
------------
 Abbott
 Alexander
 Allison
 Alston
 Arnold
 Atkinson
 Avila
 Bailey
 Baldwin
 Ball
 Barnes
 Barnett
 Barrera
 Barry
 Battle
 Bauer
 Beard
 Beasley
 Bell
 Bennett
 Benson
 Benton
 Black
 Blair
 Booker
```

（19）查询 contacts 表和 employees 表中的所有 last_name，并以 last_name 升序显示。

实现要求：保留重复+UNION ALL（必须）+其它查询方法（如果找到）

SELECT last_name FROM sales.contacts UNION ALL SELECT last_name FROM sales. employees ORDER BY last_name ASC;

```
sales=> SELECT last_name FROM sales.contacts UNION ALL SELECT last_name FROM sales. employees ORDER BY last_name ASC
;
 last_name
------------
 Abbott
 Alexander
 Allison
 Alston
 Arnold
 Atkinson
 Atkinson
 Avila
 Bailey
 Baldwin
 Ball
 Barnes
 Barnett
 Barnett
 Barrera
 Barry
 Battle
 Bauer
 Beard
 Beasley
 Bell
```

- INTERSECT 的使用

（20）查询同时出现在 contacts 表和 employees 表中的所有 last_name。

实现要求：INTERSECT（必须）+其它查询方法（如果找到）

SELECT last_name FROM sales.contacts INTERSECT SELECT last_name FROM sales.employees;

```
sales=> SELECT last_name FROM sales.contacts INTERSECT SELECT last_name FROM sales. employees;
 last_name
-----------
 Stone
 Webb
 Henry
 Brooks
 Flores
 Cruz
 Mason
 Simmons
 Cole
 Murray
 Wallace
 Ford
 Butler
 Woods
 Sanders
 Myers
 Robertson
 Ferguson
 Grant
 Henderson
 West
 Rose
 Ortiz
 Hayes
 Jordan
 Nichols
 Bryant
 Spencer
 Mcdonald
(29 rows)
```

- MINUS/EXCEPT 的使用

（21） 查询在产品表 products 中而不在库存表 inventories 中的产品号 product_id。

实现要求：MINUS/EXCEPT（必须）+其它查询方法（如果找到）

SELECT product_id FROM sales.products MINUS SELECT product_id FROM sales.inventories;

```
sales=> SELECT product_id FROM sales.products MINUS SELECT product_id FROM sales.inventori
 product_id
------------
        158
         86
         77
         97
         16
        118
        178
        153
        111
        187
         83
        179
        209
        143
         59
         10
         85
        192
        253
         48
        113
        176
```

3.实验总结

3.1 完成的工作

设计正确的 SQL 高级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）请分析归纳只能使用子查询的查询要求特点并举例说明。

一些复杂的查询需要子查询实现，特别是在处理多张表和大量数据的时候。具体例子如下：

[1] 带有聚合函数的子查询，如查询满足一定条件的最小值、最大值、平均值等聚合函数值

[2] 比较运算符中的子查询，如查询符合一定条件的元素，或查询某个表中有多少个元素满足某个条件

[3] EXIST 子查询

（2）相关子查询与不相关子查询的区别？试举例说明哪些情形下使用相关子查询，哪些情形下相关子查询可替换为一般查询实现？

（说明：一般查询是指不一定必须使用子查询的查询）

区别：

子查询是一个嵌套在另一个查询中的完整 SELECT 语句。相关子查询和不相关子查询都是子查询的一种，区别在于它们与外部查询之间的数据依赖关系。

不相关子查询是独立于主查询的子查询。子查询本身完全独立于外部查询，并且在子查询运行时不需要访问父查询中的任何数据。对于每个父查询记录，这种子查询都会返回相同的结果

相关子查询，则依赖于外部查询。与不相关查询不同，大多数时候，相关子查询需要访问外部查询（即父查询）的一些数据。对于每个父查询，返回的结果可能不一样。

相关子查询适用于那些需要对外部结果进行进一步的限制或筛选的场景，例如限制条件的细化或进一步的聚合计算。使用相关子查询可以让查询变得更加灵活，可以将多步操作合并成一步完成，提高查询效率与代码可读性。

如果将相关子查询转换成一般查询，可以通过加入子查询中使用的表来转化相关子查询。需要先根据子查询的语句重新构建一个基于派生表的查询语句，把子查询的数据依赖关系显式地表达出来。

（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,使用了 JOIN 语句来关联两张表,ON 子句指定了关联条件，如 palette_a 表和 palette_b 表的 color 列相同。

使用了 LEFT JOIN 连接两个表，并通过 ON 子句指定连接条件为 customer_id 相等。

使用了 GROUP BY 子句对结果进行分组，得到一个派生表。

使用了 EXISTS 子查询来检查 orders 表中是否存在该顾客的订单，如果存在，则选择该顾客的名字。

使用 NOT EXISTS 子查询来查找没有任何订单的客户。

使用 ANY 子查询来查找定价高于某个阈值的产品。

使用 ALL 子查询来查找定价高于某个阈值的所有产品。

使用 ALL 子查询来选择定价低于所有产品种类的平均定价的产品。

使用 UNION 运算符来联合两个表中的 last_name 记录。

使用 UNION ALL 运算符，保留了两张表中所有相同和不同的 last_name 记录。

使用 INTERSECT 运算符，连接两个 SELECT 语句并返回它们的交集。

使用 MINUS 运算符,连接两个 SELECT 语句并返回它们的差集。

3.3 遇到的困难及解决方法

无。