

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： | 2024.4.13—4.13 |
| 实验地点： | 文宣楼B313 & 课后其他地点 |
| 提交日期： | 2024.4.13 |
|  | |
| 学号： | 37220222203791 |
| 姓名： | 魏一诺 |
| 专业年级： | 软工2022级 |
| 学年学期： | 2023-2024学年第二学期 |

1. 实验目的
2. 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
3. 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

1. 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
2. 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。

1. 实验内容和步骤

**（1）连接到数据库**

步骤如下：

cd /opt/software/openGauss/script/ //步骤 1 在数据库主节点服务器上，切换至omm操作系统用户环境

su - omm

gs\_om -t status //步骤 2 查看服务是否启动

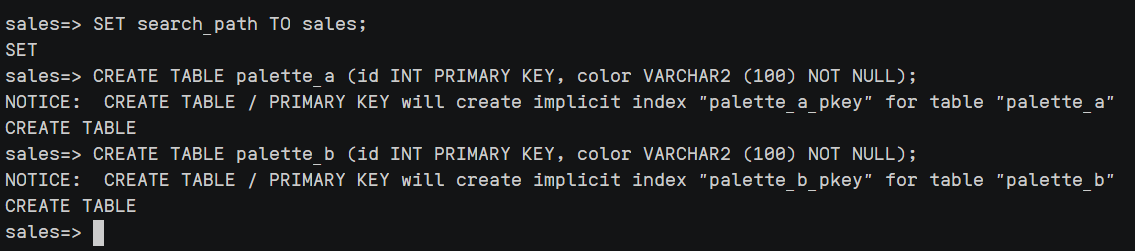
gs\_om -t start //步骤 3 启动数据库服务

gsql -d sales -p 26000 -U wyn -W Bigdata@123 -r //连接到数据库

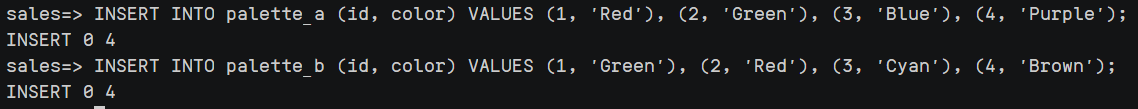
CREATE SCHEMA Sales AUTHORIZATION wyn;

SET search\_path TO sales;

**（2）创建两张表 palette\_a 和 palette\_b（结构相同，但表名不同，color 为颜色）**



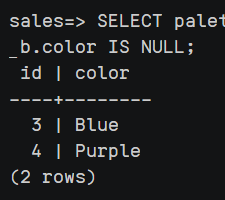
**（3）添加样例数据**



**(5)查询 palette*a 表中颜色不出现在 palette*b 表中的 id 和颜色。**

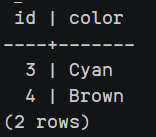
实现要求：左外连接（必须）+其它查询方法（如果找到）

SELECT palette\_a.id,palette\_a.color FROM palette\_a LEFT JOIN palette\_b ON palette\_a.color = palette\_b.color WHERE palette\_b.color IS NULL;



**（6)查询palette\_b表中颜色不出现在palette\_a表中的id和颜色**

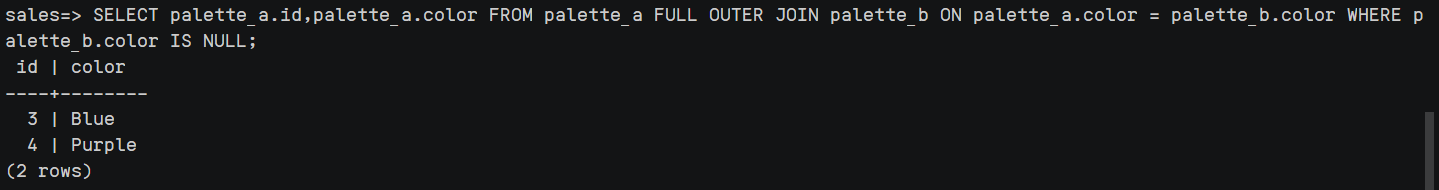
实现要求：右外连接（必须）+其它查询方法（如果找到）

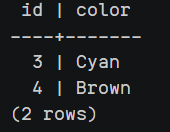


### (7) 查询（5）或（6）两种情况的信息（用（全）外连接）。

SELECT palette\_a.id,palette\_a.color FROM palette\_a FULL OUTER JOIN palette\_b ON palette\_a.color = palette\_b.color WHERE palette\_b.color IS NULL;

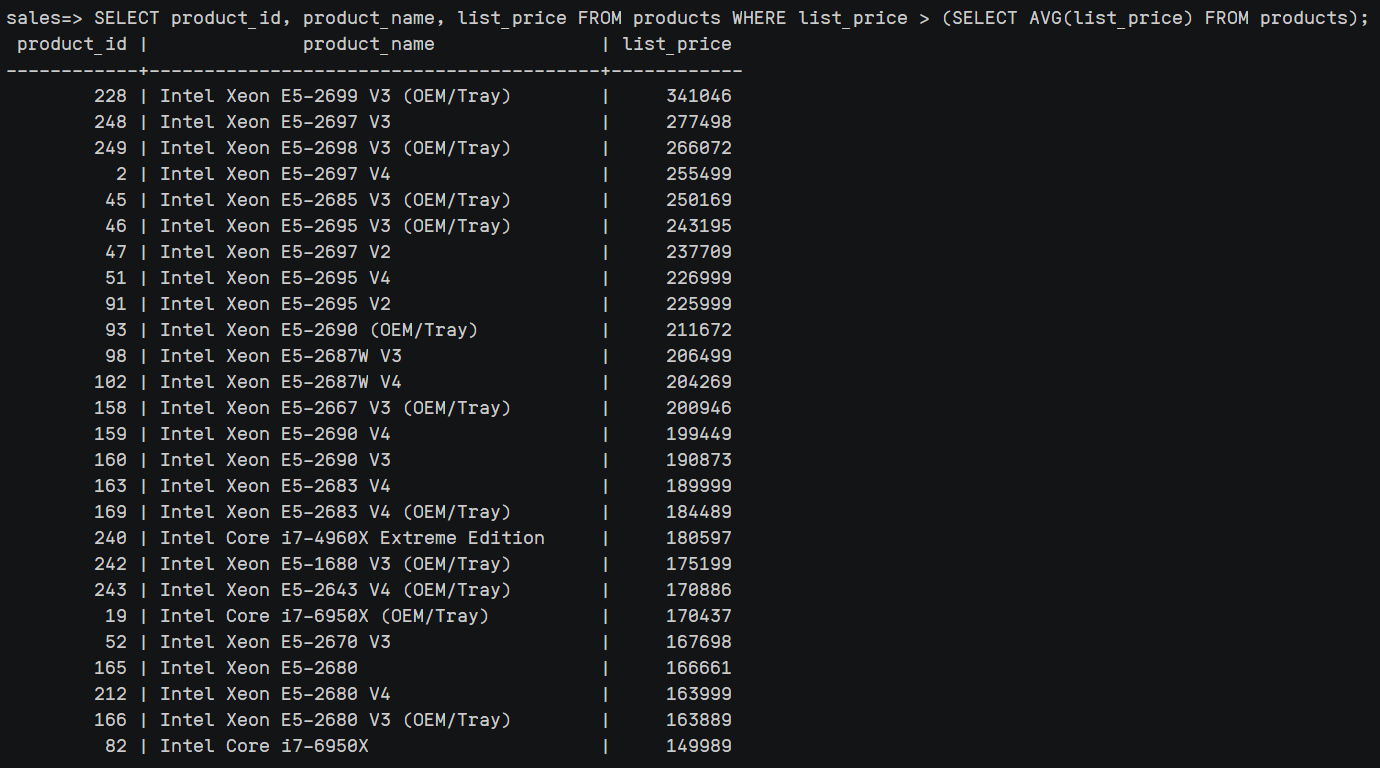
SELECT palette\_b.id,palette\_b.color FROM palette\_b FULL OUTER JOIN palette\_a ON palette\_b.color = palette\_a.color WHERE palette\_a.color IS NULL;





### (8)查询产品表 products 中的 productid, productname, listprice 信息，要求产品定价listprice 大于其平均定价 list\_price。

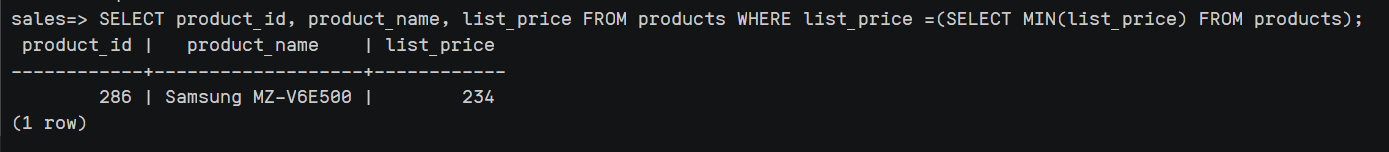
SELECT product\_id, product\_name, list\_price FROM products WHERE list\_price > (SELECT AVG(list\_price) FROM products);



### (9）查询产品表 products 中最便宜产品的 productid, productname, list\_price。

### 指令如下：

SELECT product\_id, product\_name, list\_price FROM products WHERE list\_price =(SELECT MIN(list\_price) FROM products);

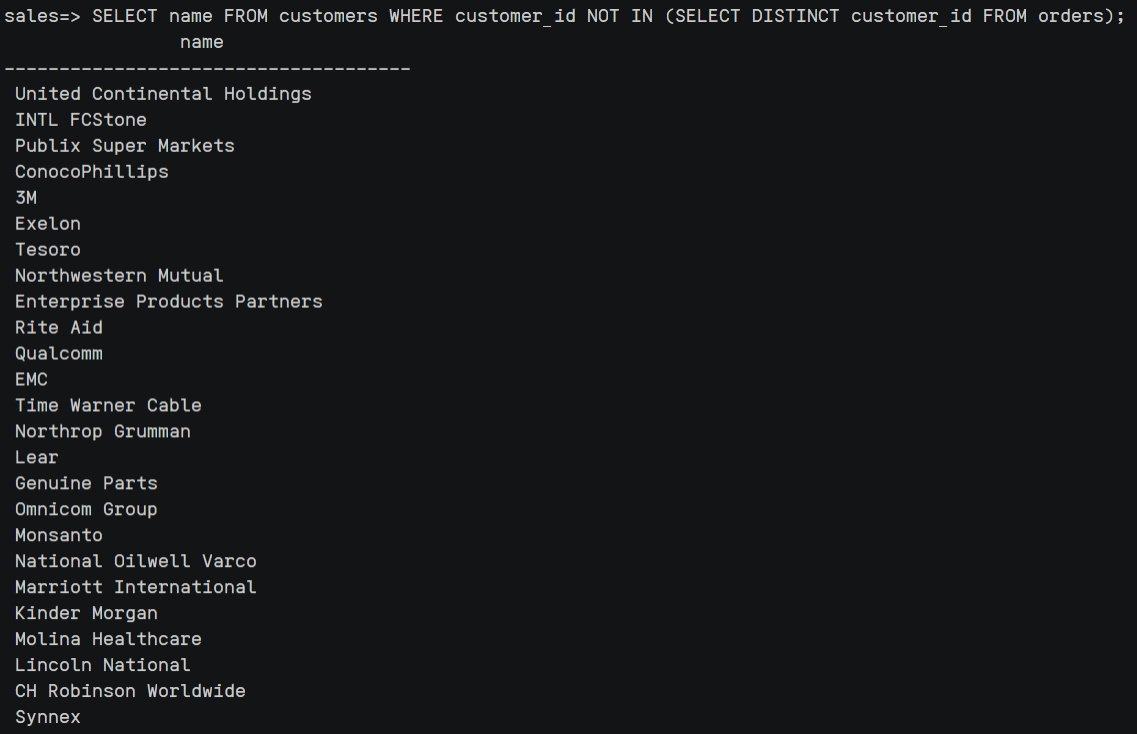


**(10) 查询没有一个订单的顾客姓名。**

**指令如下：**

**SELECT name FROM customers WHERE customer\_id NOT IN (SELECT DISTINCT customer\_id FROM orders);**

**结果如下：**

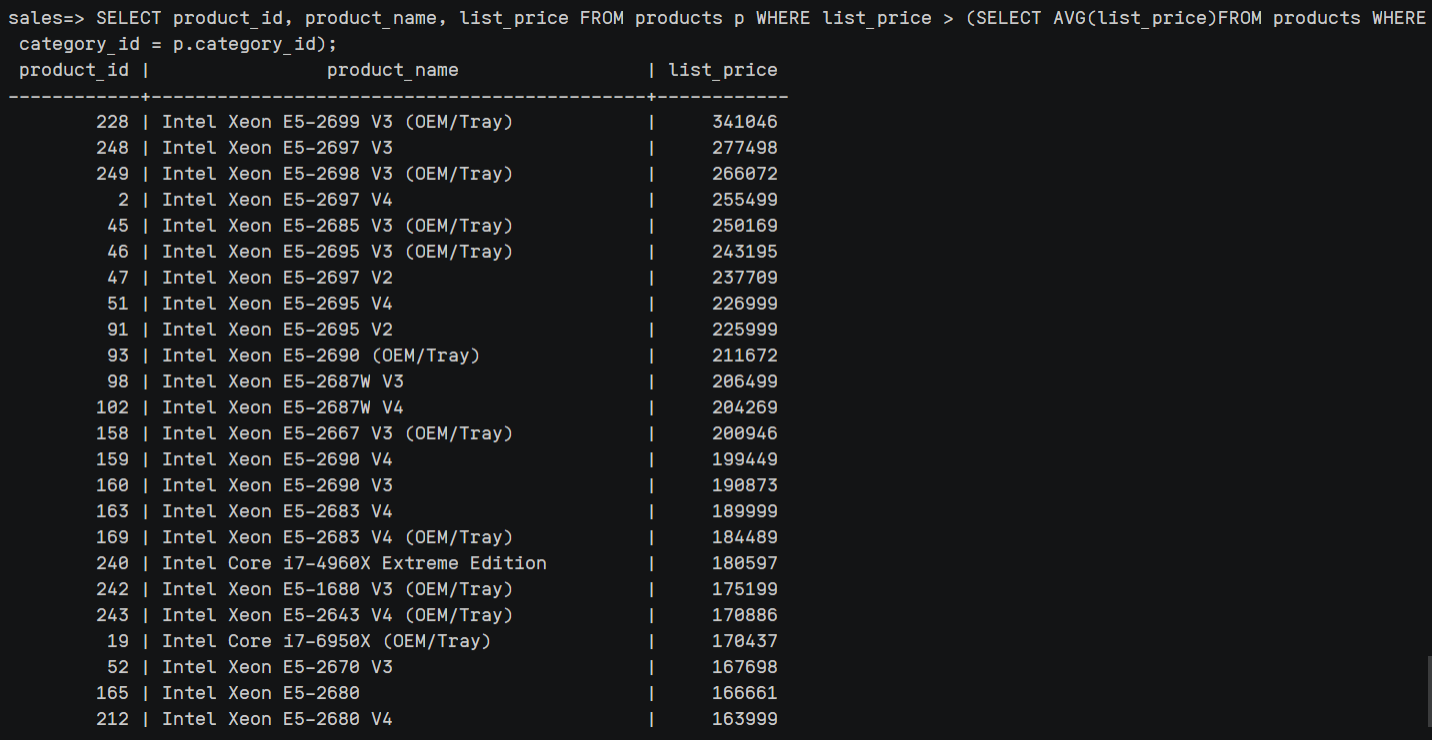
****

### （11） 查询产品表 products 中产品的 productid, productname, listprice，要求产品定价listprice 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）

指令和结果如下：

SELECT product\_id, product\_name, list\_price FROM products p WHERE list\_price > (SELECT AVG(list\_price)FROM products WHERE category\_id = p.category\_id);



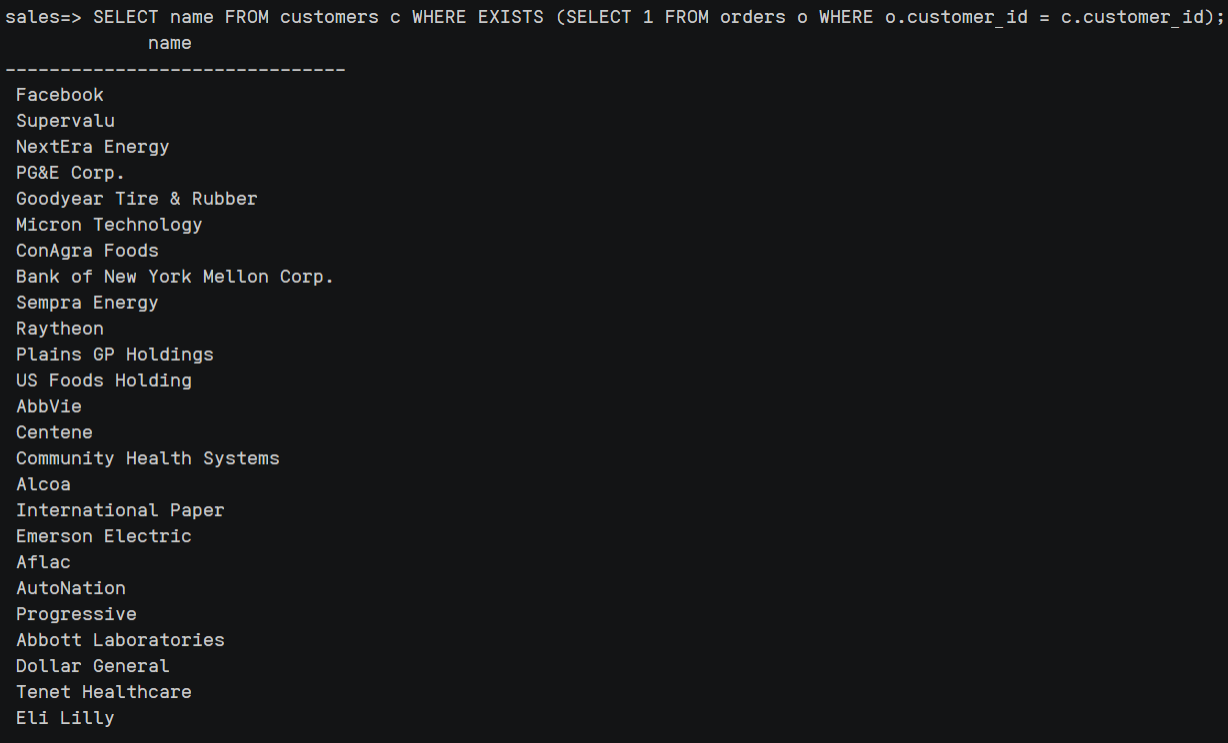
### （12）查询有订单 order 的所有顾客 customer 姓名（查询涉及 customers 表和 orders 表）。

实现要求：使用 EXISTS（必须）+其它查询方法（如果找到）

指令如下：

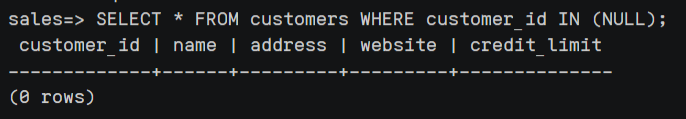
SELECT name FROM customers c WHERE EXISTS (SELECT 1 FROM orders o WHERE o.customer\_id = c.customer\_id);

结果如下：



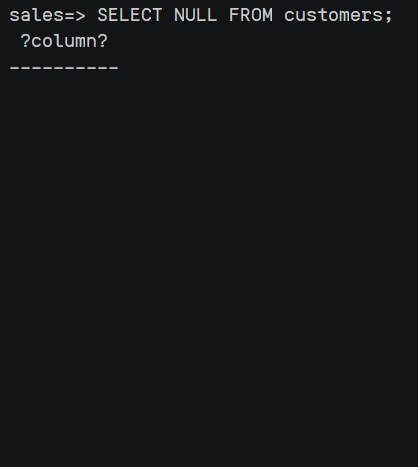
### （13）执行以下三条与 NULL 相关的语句，观察各自执行的结果，能否从中得出某些初步结论？

执行结果与初步结论：

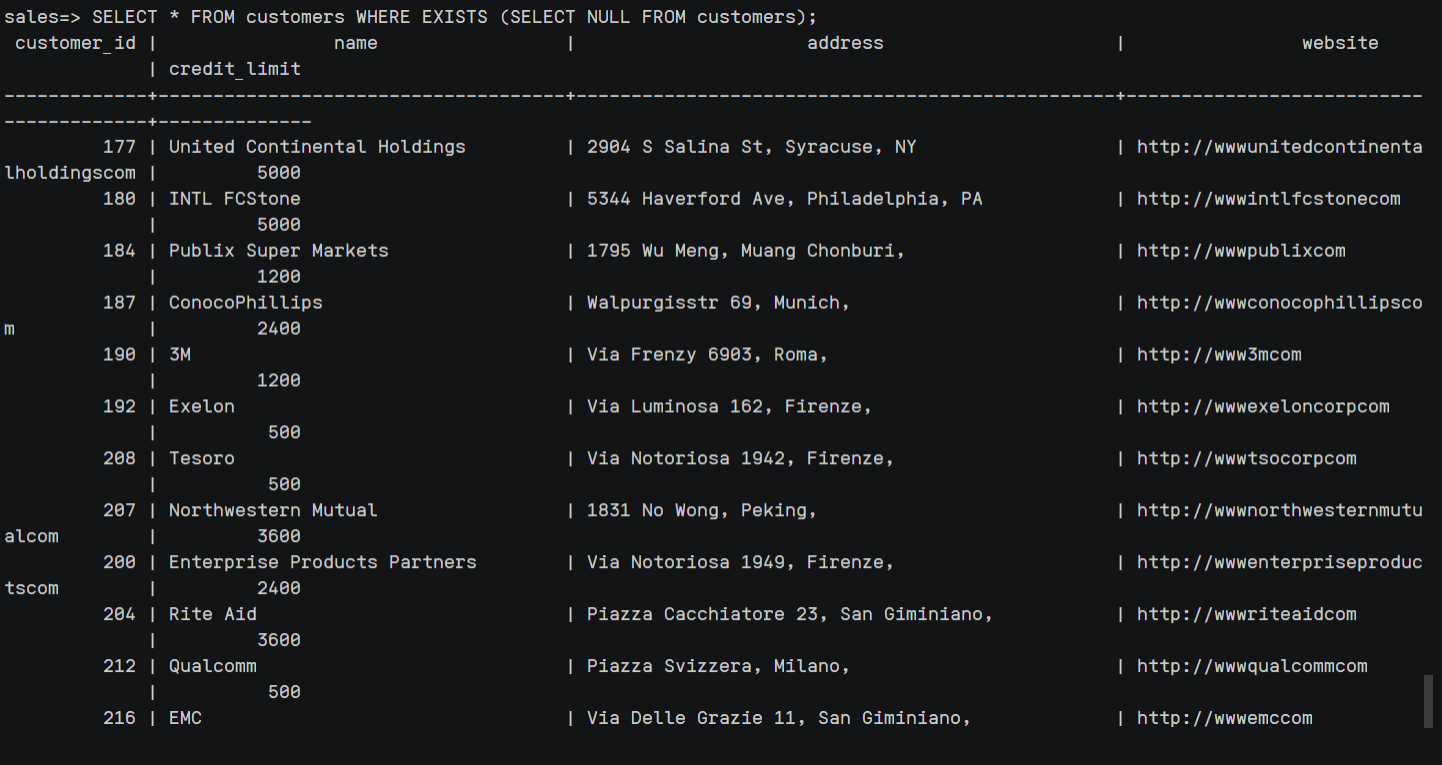


SELECT \* FROM customers WHERE customer\_id IN (NULL);

--搜寻 customers表中 customer\_id为空的记录，结果是没有 所以返回了0行



SELECT NULL FROM customers;--在从名为 "customers" 的表中选择 NULL 值。数据库管理系统尝试根据表中的行数来生成相应数量的结果行，每行都包含一个 NULL值。



SELECT \* FROM customers WHERE EXISTS (SELECT NULL FROM customers);--EXISTS会筛选符合后面条件的行 但写了NULL就相当于没有做这个筛选，因此输出了所有行

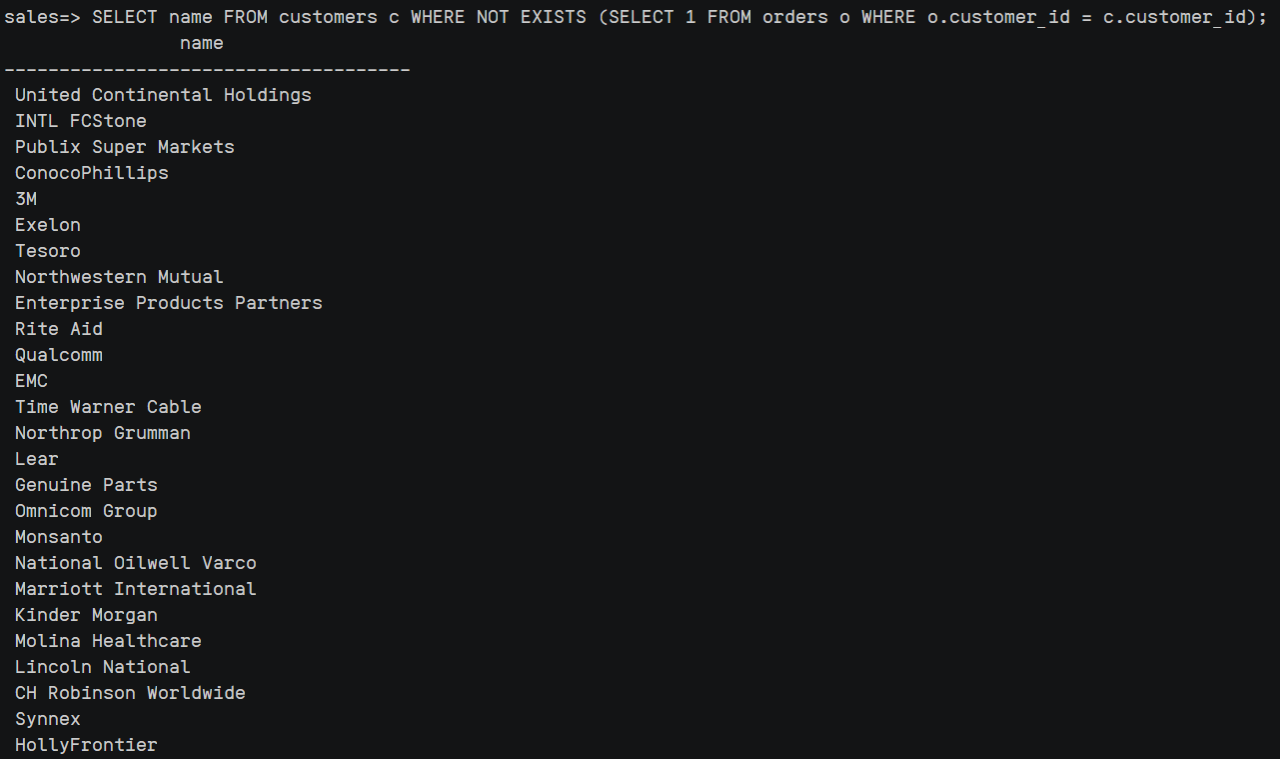
### （14）找出所有没有订单的顾客姓名（查询涉及 customers 表和 orders 表）。

实现要求：使用 NOT EXISTS（必须）+其它查询方法（如果找到）

代码如下：

SELECT name FROM customers c WHERE NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer\_id = c.customer\_id);

结果展示：

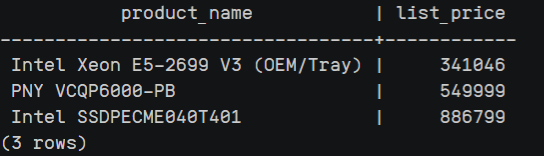


### （15）查询产品表 products 中的产品名 productname 和定价 listprice，要求其定价高于产品种类 1 中的任何产品定价。实现要求：ANY（必须）+其它查询方法（如果找到）

代码如下：

SELECT product\_name, list\_price FROM products AS p1 WHERE NOT EXISTS (SELECT 1 FROM products AS p2 WHERE p2.category\_id = 1 AND p1.list\_price < ANY (SELECT list\_price FROM products WHERE category\_id = 1));

结果展示：

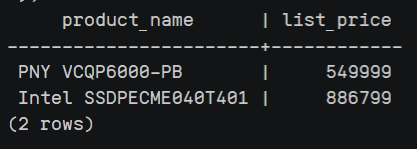


### （16）查询产品表 products 中的产品名 productname 和定价 listprice，要求其定价高于产品种类 1 中的所有定价。

指令如下：

SELECT product\_name, list\_price FROM products WHERE list\_price > ALL (SELECT list\_price FROM products WHERE category\_id = 1);

结果展示：



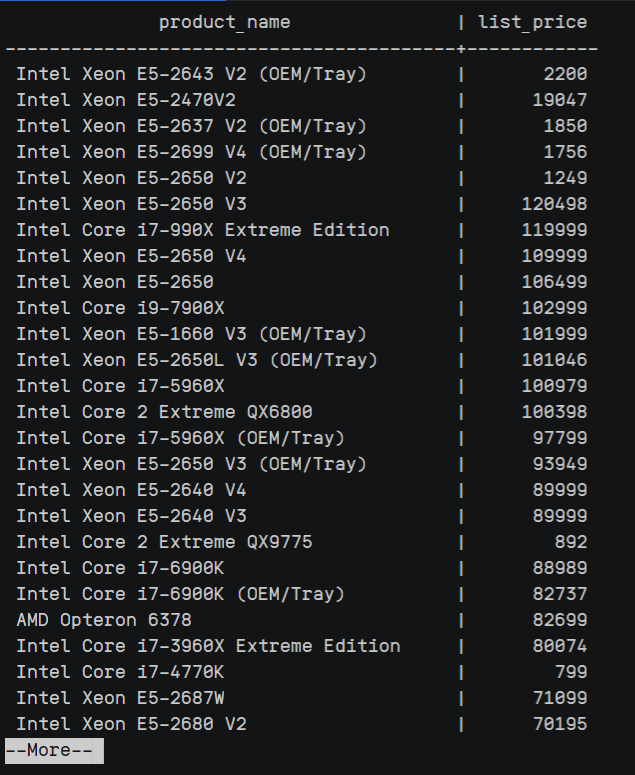
### （17）查询产品表 products 中的产品名 productname 和定价 listprice，要求其定价低于产品种类的所有平均定价。

实现要求：ALL（必须）+其它查询方法（如果找到）

指令如下：

SELECT product\_name, list\_price FROM products p WHERE list\_price < ALL(SELECT AVG(list\_price)FROM products WHERE category\_id = p.category\_id);

结果展示：



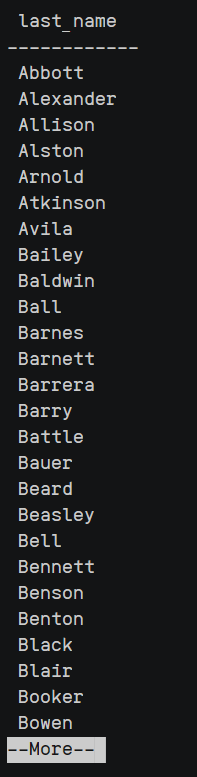
### （18）查询 contacts 表和 employees 表中的所有 lastname，并以 lastname 升序显示。

实现要求：去重+UNION（必须）+其它查询方法（如果找到）

指令如下：

SELECT DISTINCT last\_name FROM contacts UNION SELECT DISTINCT last\_name FROM employees ORDER BY last\_name ASC;

结果展示：



### （19）查询 contacts 表和 employees 表中的所有 lastname，并以 lastname 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）

指令如下：

SELECT last\_name FROM (SELECT last\_name FROM contacts UNION ALL SELECT last\_name FROM employees) AS combined\_data ORDER BY last\_name ASC;

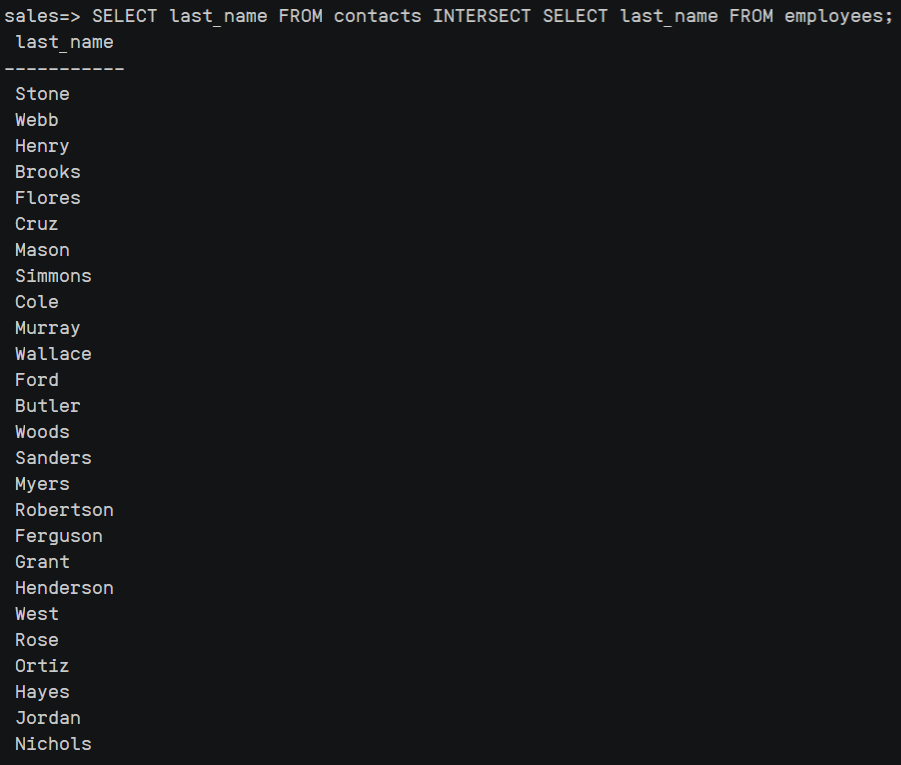
结果展示：



### （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 contacts INTERSECT SELECT last\_name FROM employees;



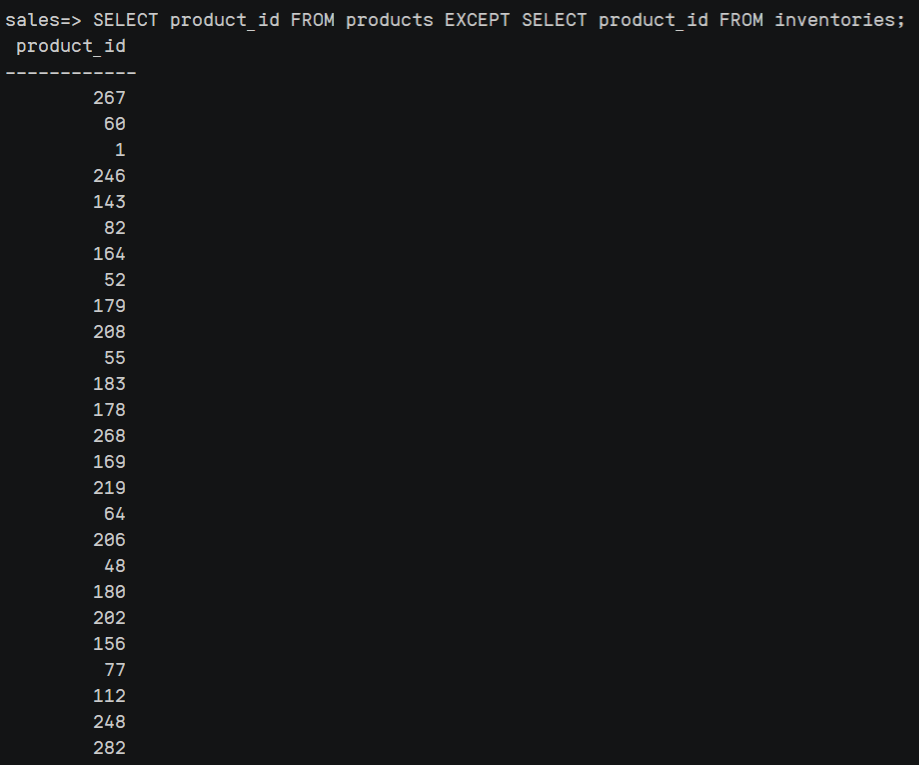
### （21） 查询在产品表 products 中而不在库存表 inventories 中的产品号 product\_id。

实现要求：MINUS/EXCEPT（必须）+其它查询方法（如果找到）

指令如下：

SELECT product\_id FROM products EXCEPT SELECT product\_id FROM inventories;

结果如下：



实验总结

3.1 完成的工作

完成了内连接、（左右）外连接、子查询和集合查询等相关练习

3.2 对实验的认识

对这些查询方法有了更深入的理解

3.3 遇到的困难及解决方法

有因为date类型数据格式产生的错误和手误输错表名产生的错误，调整格式、改正后就解决了。