

数据库系统课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 数据高级查询 |
| 实验日期： | 2022.11.11 |
| 实验地点： | 四号楼 |
| 提交日期： | 2022.11.14 |
|  | |
| 学号： | 22920202202877 |
| 姓名： | 陈鑫蕾 |
| 专业年级： | 数媒2020级 |
| 学年学期： | 2022-2023学年第一学期 |

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

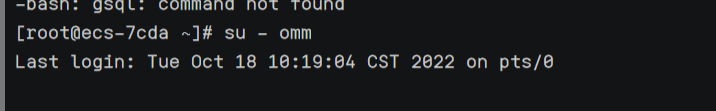
熟练掌握基于派生表的查询方法

1. 实验内容和步骤

（1）连接到数据库

步骤如下：

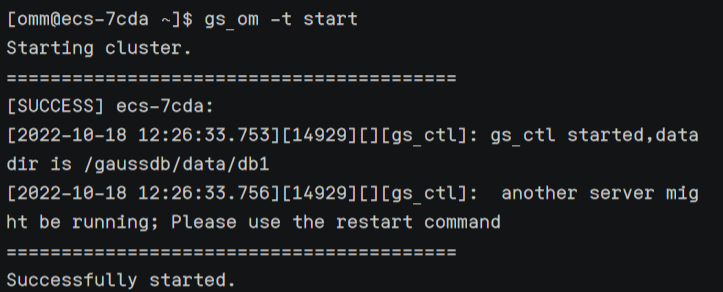
1.在数据库主节点服务器上，切换至omm操作系统用户环境。



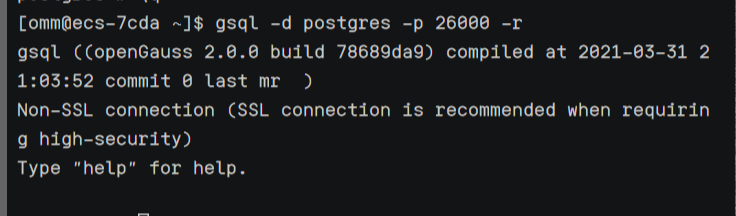
2.查看服务是否启动。



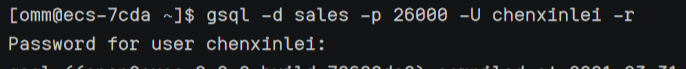
3.启动数据库服务



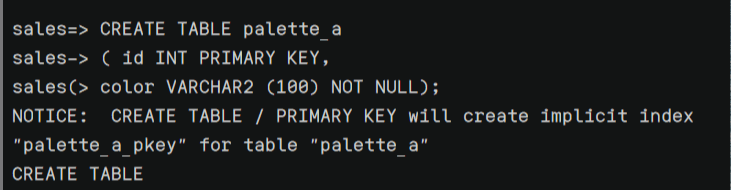
4.连接数据库

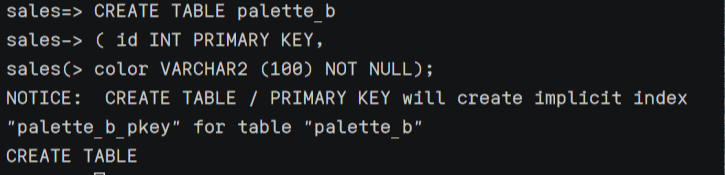


1. 使用自己创建的用户连接到此数据库



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





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

INSERT INTO palette\_a

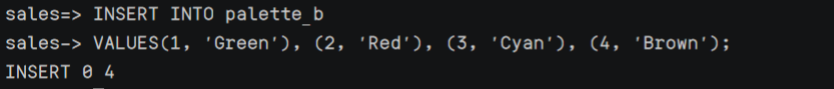
VALUES(1, 'Red'), (2, 'Green'), (3, 'Blue'), (4, 'Purple');

35(3L(6UU~F)[RWMIXV_L@A

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

INSERT INTO palette\_b

VALUES(1, 'Green'), (2, 'Red'), (3, 'Cyan'), (4, 'Brown');



1. 查询两张表中相同颜色的所有信息。

SELECT \*

FROM palette\_a,palette\_b

WHERE palette\_a.color=palette\_b.color;



1. 查询 palette\_a 表中颜色不出现在 palette\_b 表中的两张表的 id 和颜色（用左外连接）。

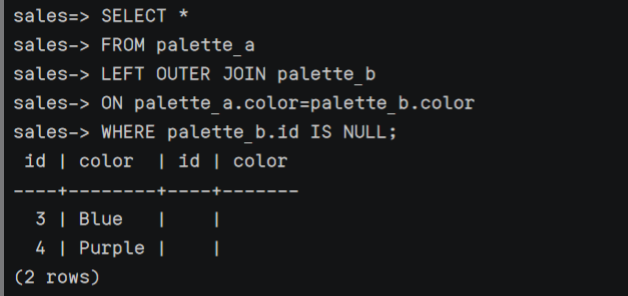
SELECT \*

FROM palette\_a

LEFT OUTER JOIN palette\_b

ON palette\_a.color=palette\_b.color

WHERE palette\_b.id IS NULL;



1. 查询 palette\_b 表中颜色不出现在 palette\_a 表中的两张表的 id 和颜色（用右外连接）

SELECT \*

FROM palette\_a

RIGHT OUTER JOIN palette\_b

ON palette\_a.color=palette\_b.color

WHERE palette\_a.id IS NULL;



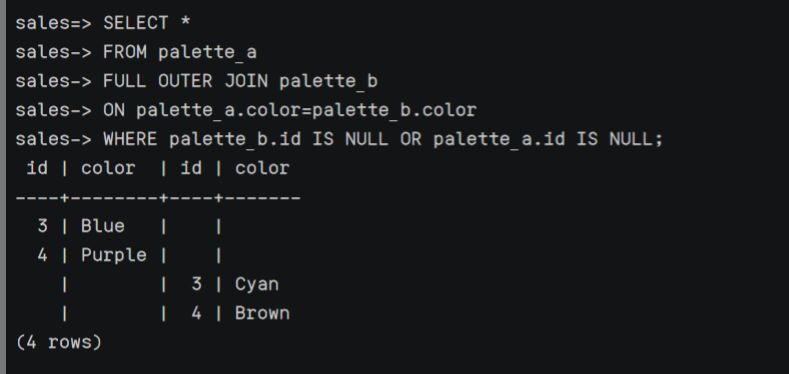
1. 查询（5）或（6）两种情况的信息（用（全）外连接）。

FROM palette\_a

FULL OUTER JOIN palette\_b

ON palette\_a.color=palette\_b.color

WHERE palette\_b.id IS NULL OR palette\_a.id IS NULL;



1. 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

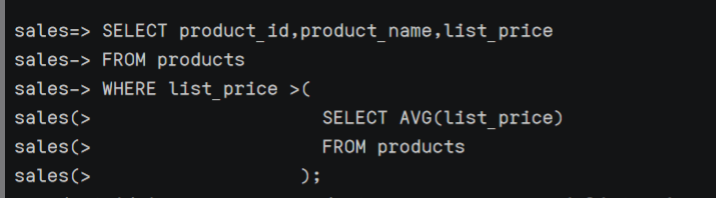
FROM products

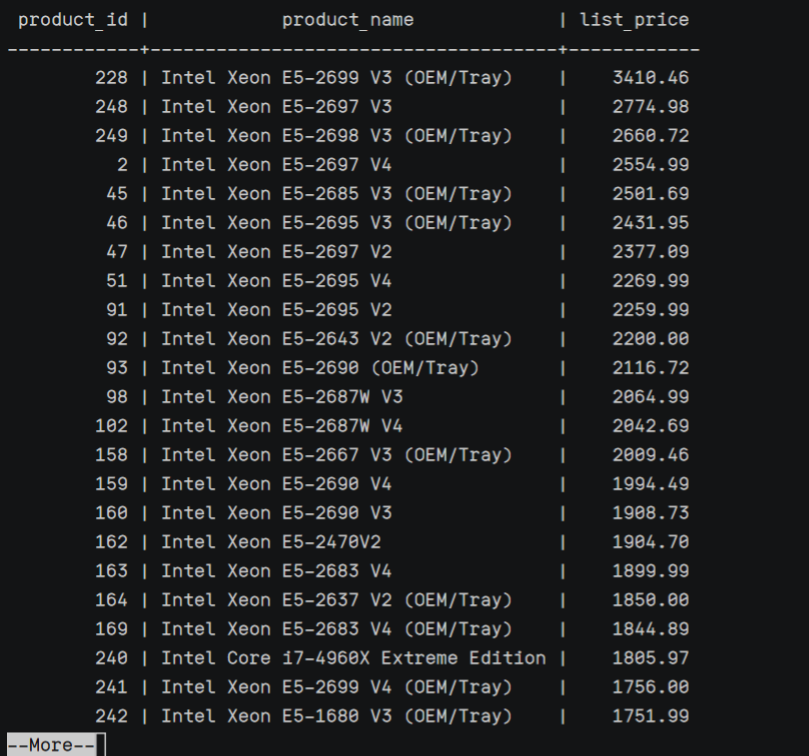
WHERE list\_price >(

SELECT AVG(list\_price)

FROM products

);







1. 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

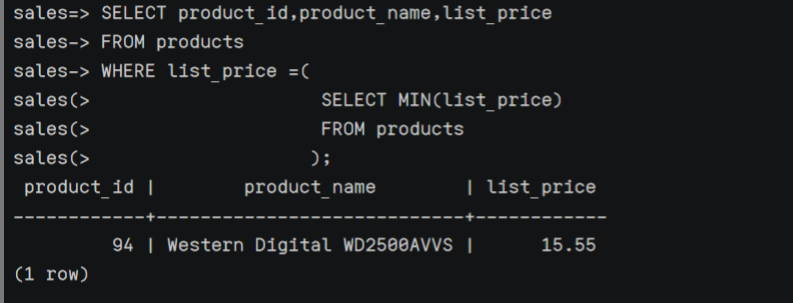
FROM products

WHERE list\_price =(

SELECT MIN(list\_price)

FROM products

);



1. 查询没有一个订单的顾客姓名。

SELECT name

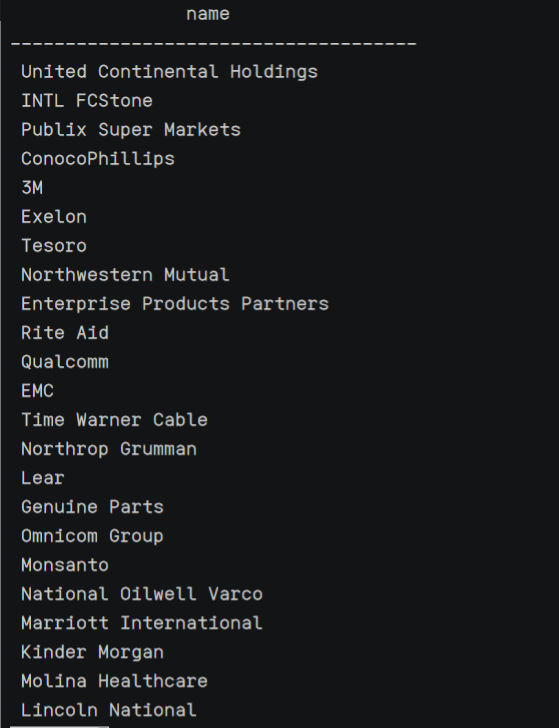
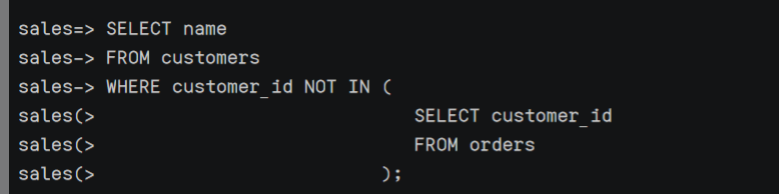
FROM customers

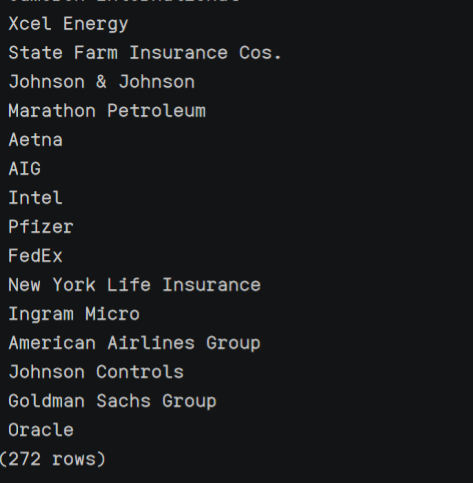
WHERE customer\_id NOT IN (

SELECT customer\_id

FROM orders

);





1. 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。

SELECT product\_id,product\_name,list\_price

FROM products x

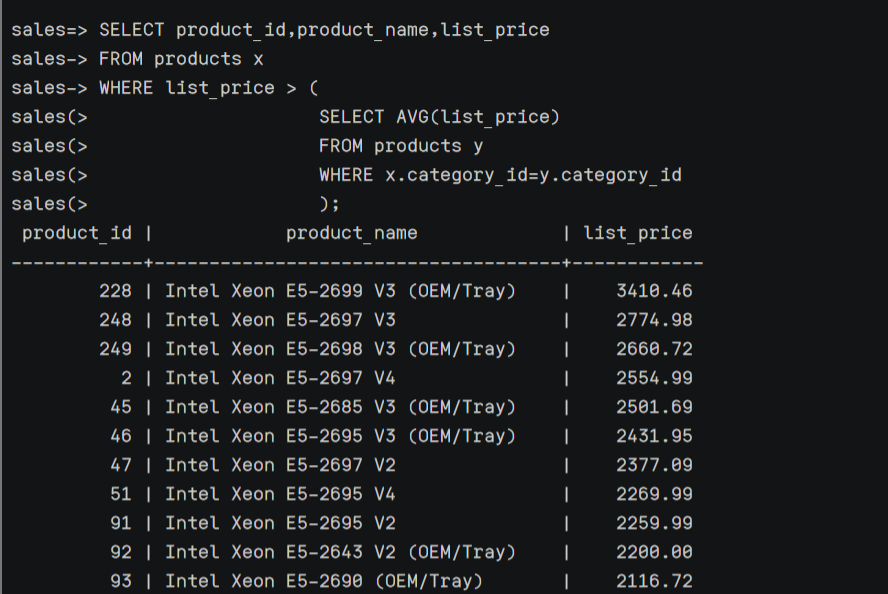
WHERE list\_price > (

SELECT AVG(list\_price)

FROM products y

WHERE x.category\_id=y.category\_id

);





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

SELECT name

FROM customers

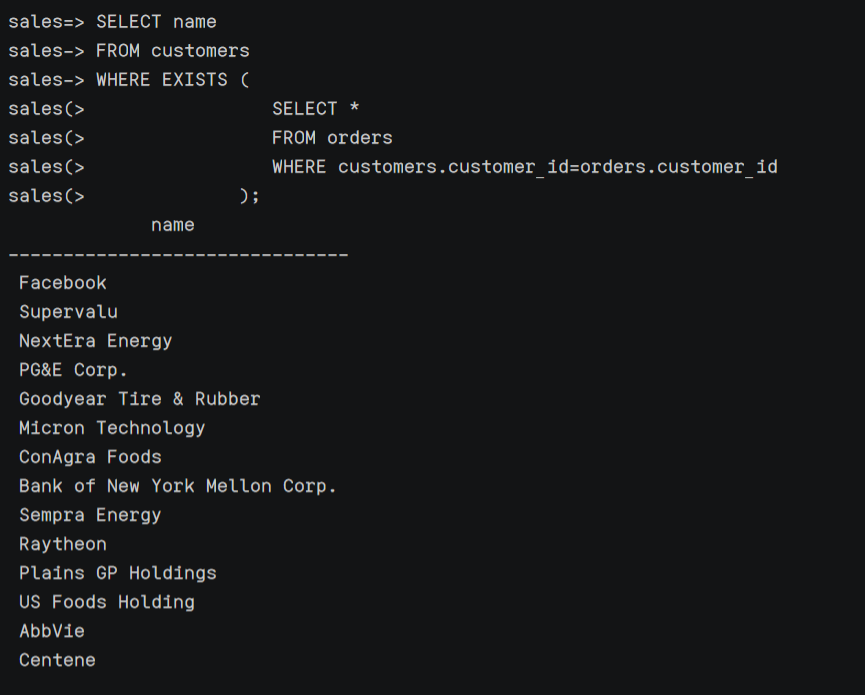
WHERE EXISTS (

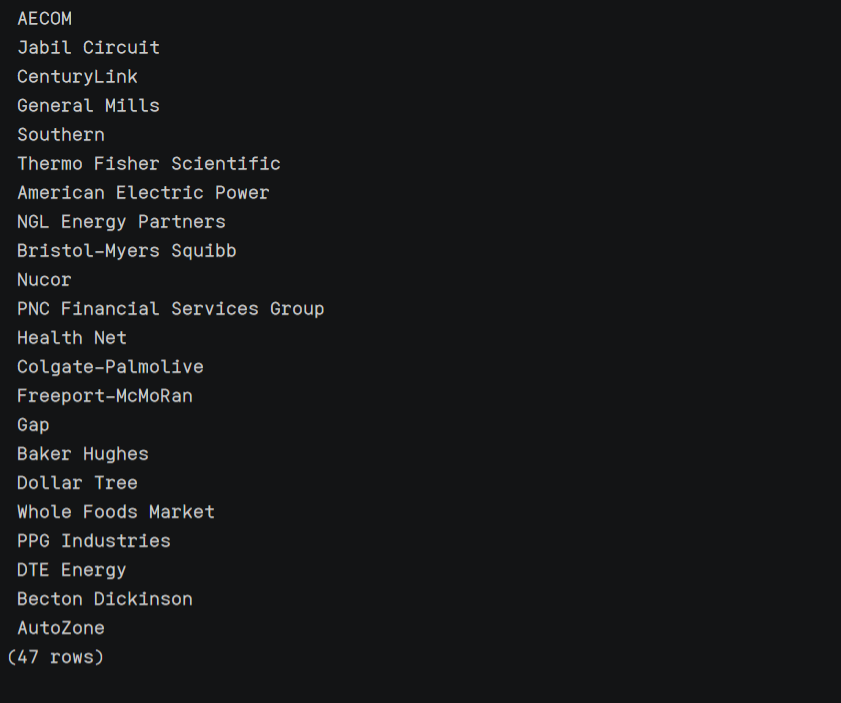
SELECT \*

FROM orders

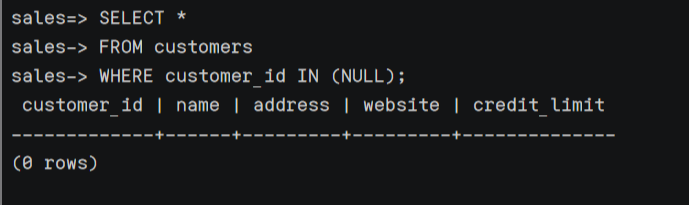
WHERE customers.customer\_id=orders.customer\_id

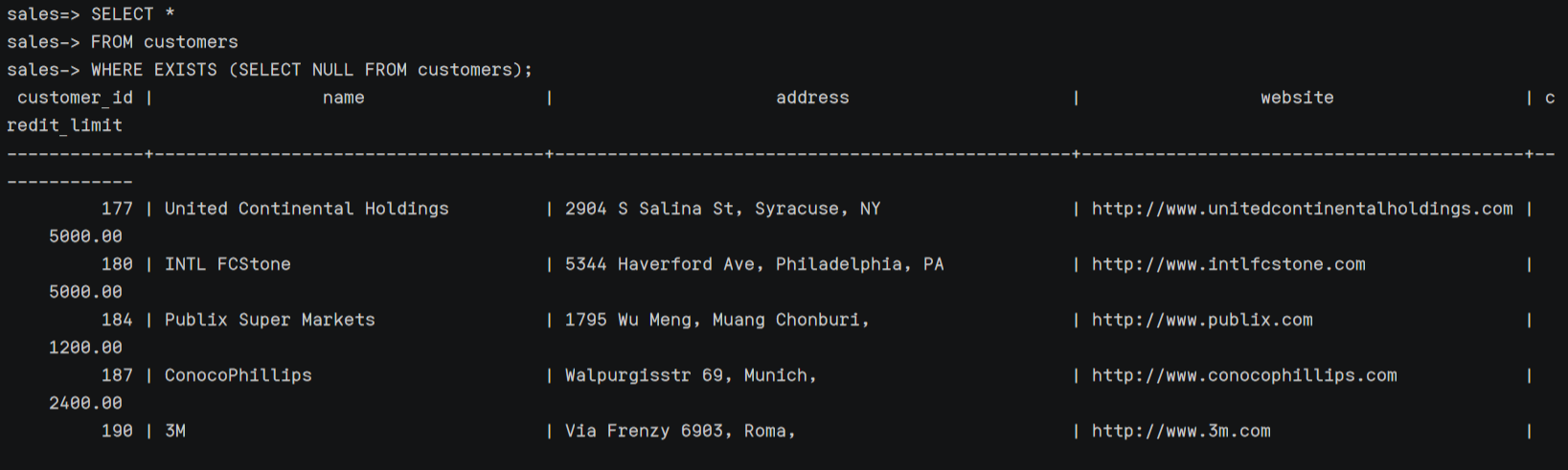
);





1. 执行以下两条语句，观察有何不同，能否得出某些初步结论





第一条查询的是id为null的情况，但是id为主码，不可能为空，所以返回0条结果；第二条是判断exists后的情况，SELECT NULL FROM table返回值为1；

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

SELECT name

FROM customer

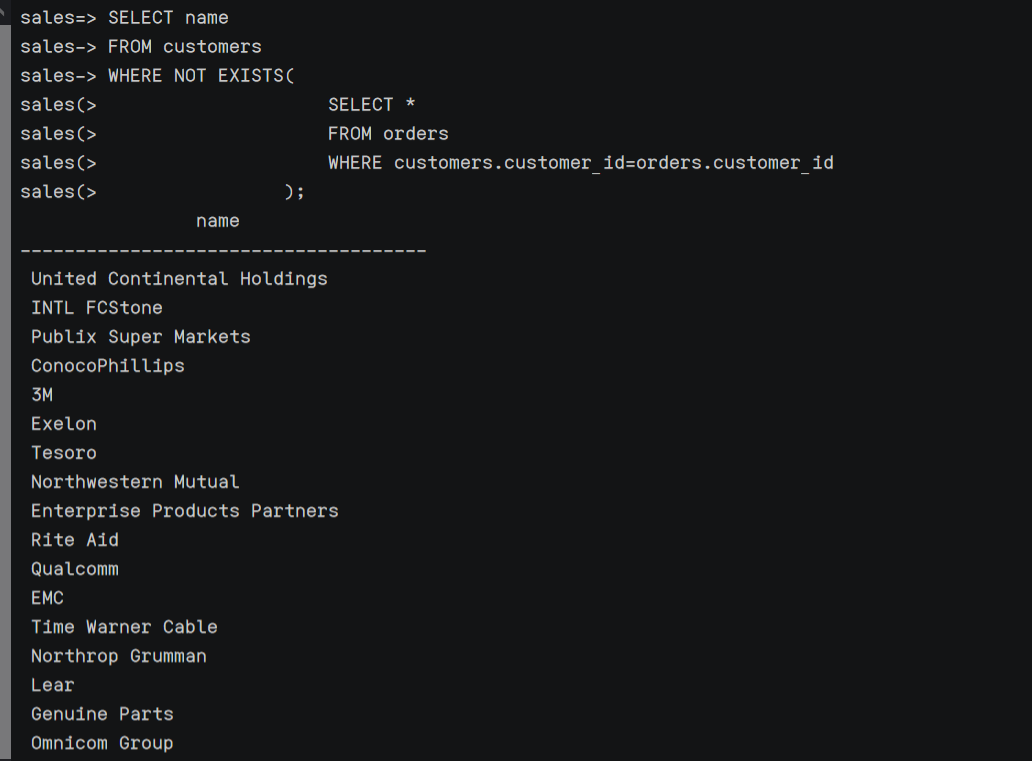
WHERE NOT EXISTS(

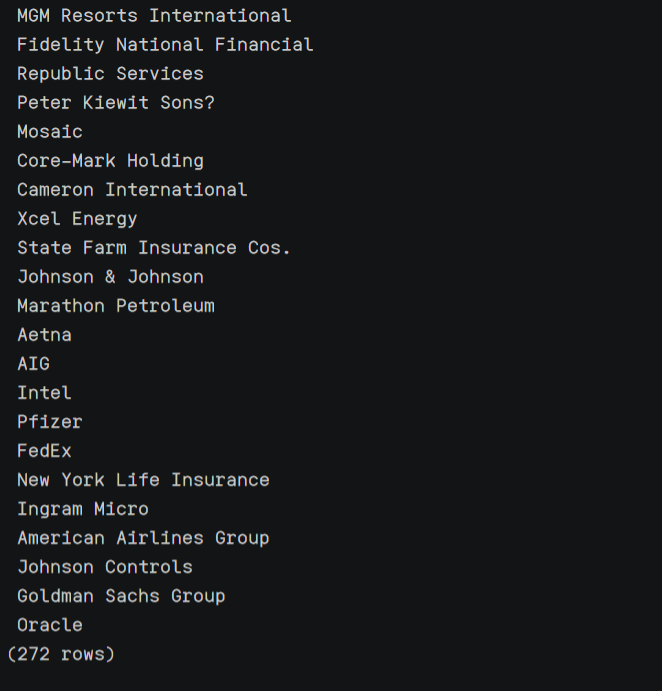
SELECT \*

FROM orders

WHERE customers.customer\_id=orders.customer\_id

);





1. 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。

SELECT product\_name,list\_price

FROM products

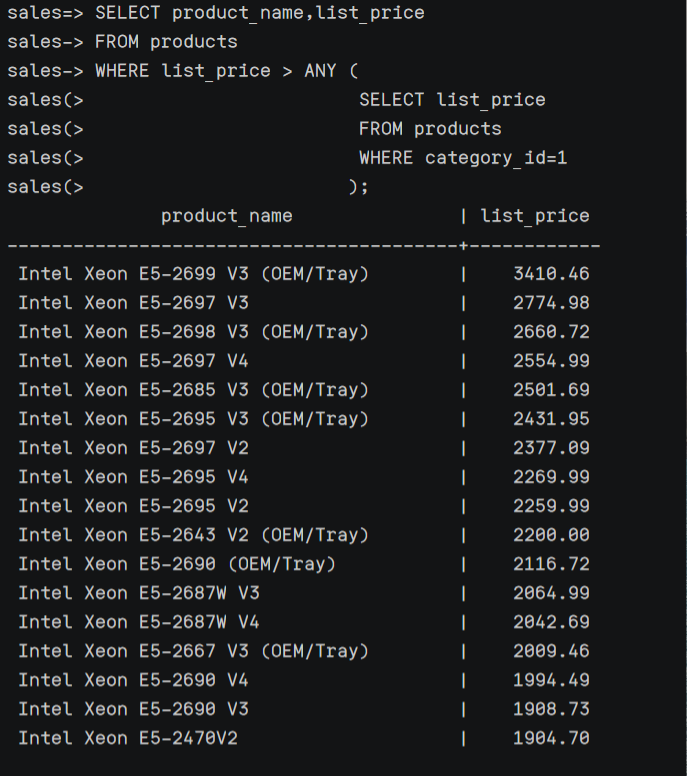
WHERE list\_price > ANY (

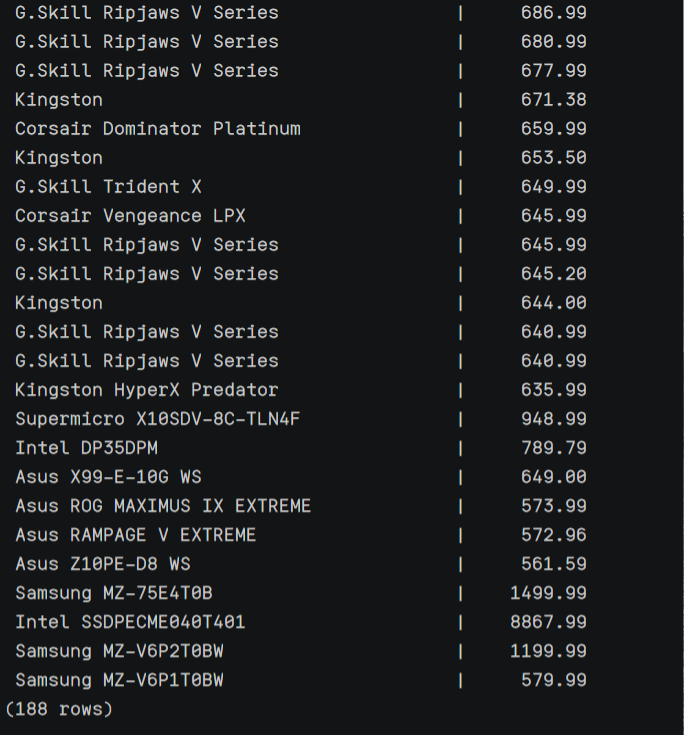
SELECT list\_price

FROM products

WHERE category\_id=1

);





1. 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 products

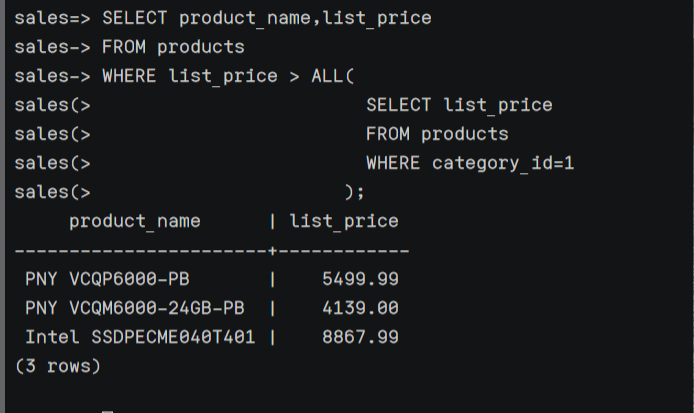
WHERE list\_price > ALL(

SELECT list\_price

FROM products

WHERE category\_id=1

);



1. 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。

SELECT product\_name,list\_price

FROM products

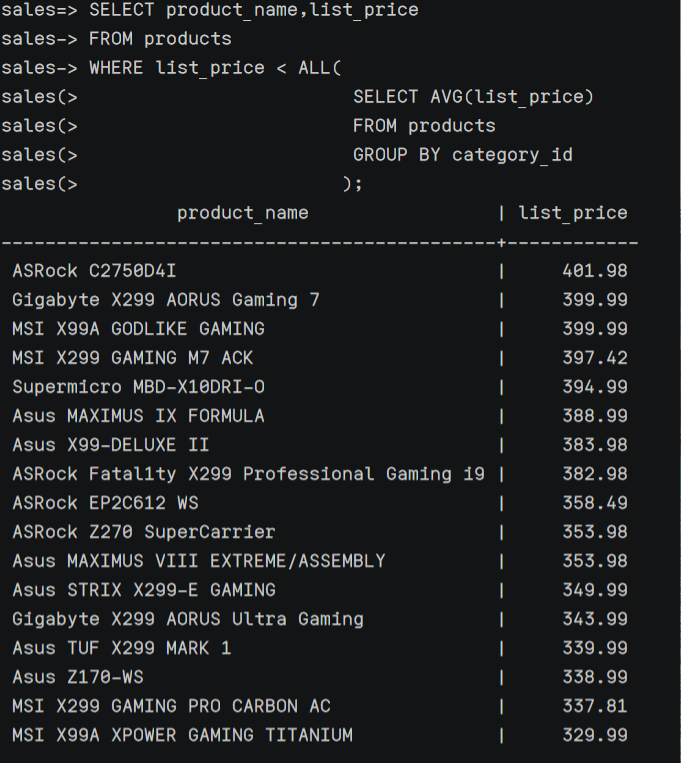
WHERE list\_price < ALL(

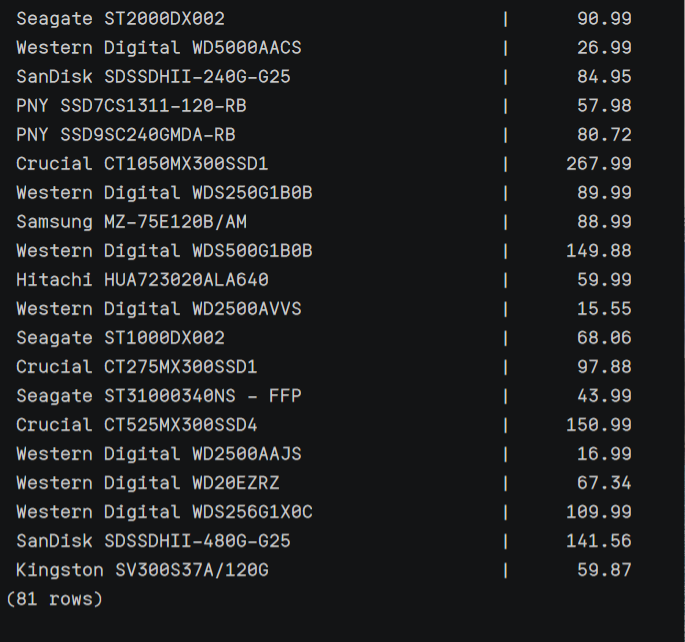
SELECT AVG(list\_price)

FROM products

GROUP BY category\_id

);





1. 查询 contacts 表和 employees 表中的所有 last\_name，并以 last\_name 升序显示。

SELECT last\_name

FROM(

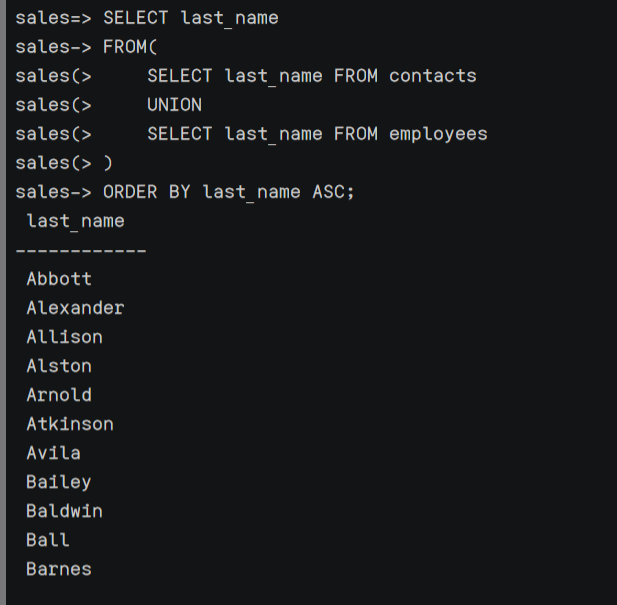
SELECT last\_name FROM contacts

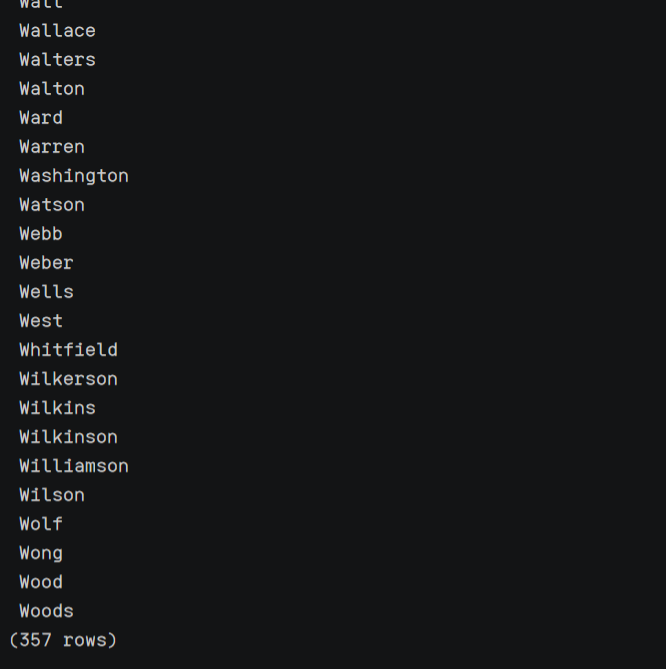
UNION

SELECT last\_name FROM employees

)

ORDER BY last\_name ASC;





1. 查询 contacts 表和 employees 表中的所有 last\_name，并以 last\_name 升序显示

SELECT last\_name

FROM(

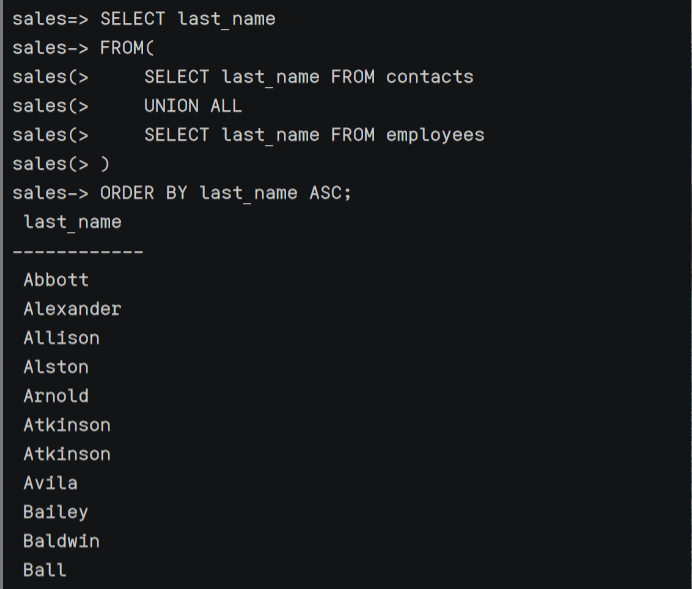
SELECT last\_name FROM contacts

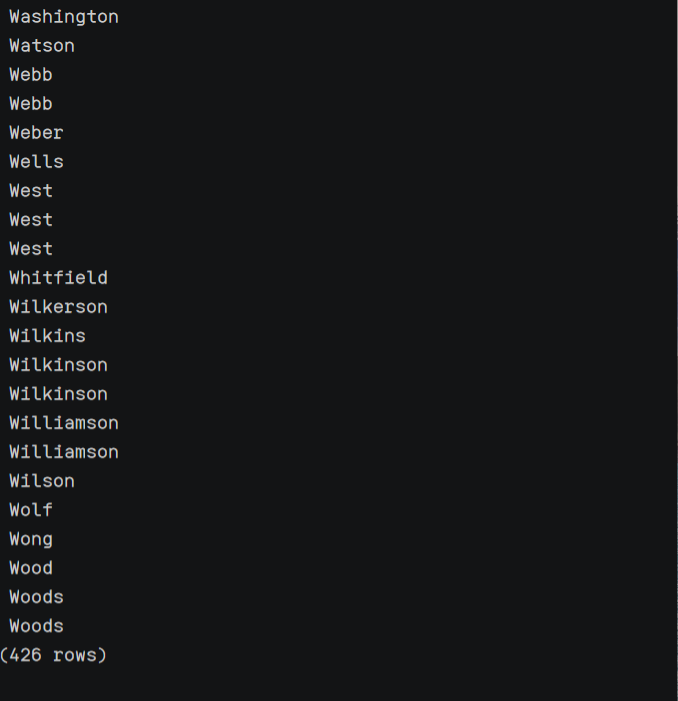
UNION ALL

SELECT last\_name FROM employees

)

ORDER BY last\_name ASC;





1. 查询同时出现在 contacts 表和 employees 表中的所有 last\_name。

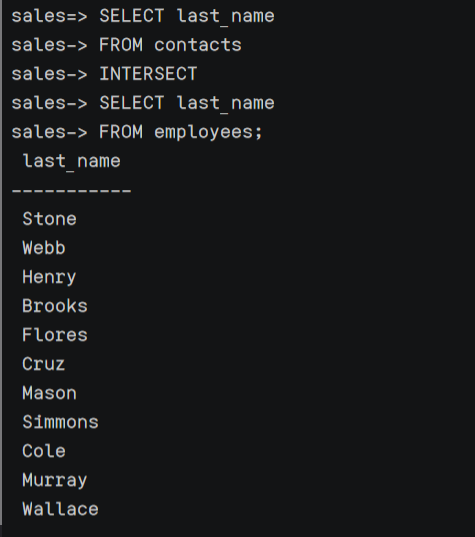
SELECT last\_name

FROM contacts

INTERSECT

SELECT last\_name

FROM employees;





1. 查询在产品表 products 中而不在库存表 inventories 中的产品号 product\_id。

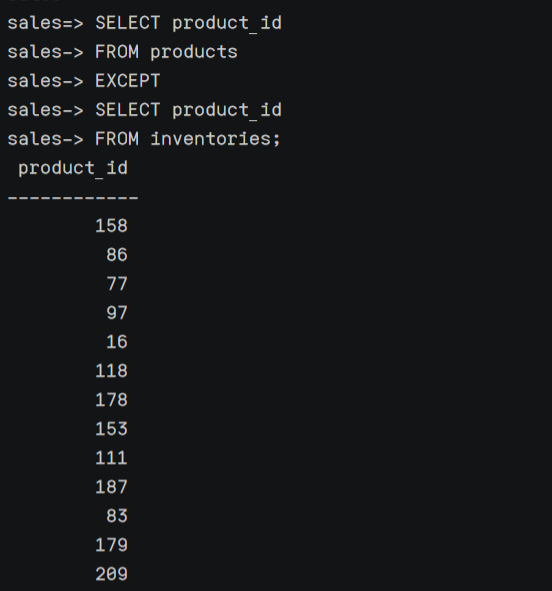
SELECT product\_id

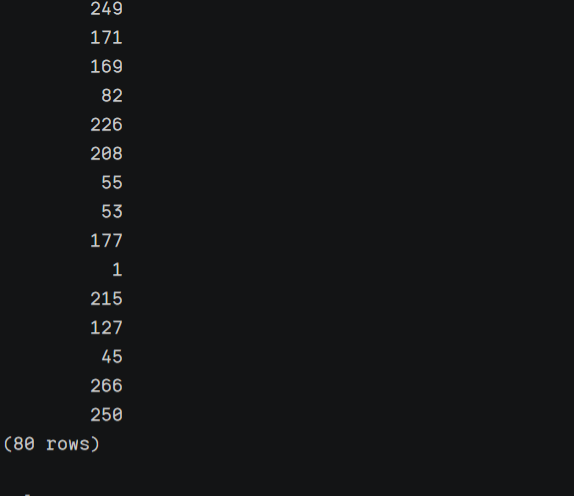
FROM products

EXCEPT

SELECT product\_id

FROM inventories;





1. 思考：

 什么类型的查询只能用子查询实现？试举例说明。

当查询的数据不能直接获取时，比如查询没有一个订单的顾客姓名

 相关子查询与不相关子查询的区别？什么情形下使用相关子查询？如何将相关子查询转换成一般查询？（说明：一般查询指不一定必须使用子查询的查询）

不相关子查询中子查询的条件不依赖于父查询；相关子查询中的子查询依赖父查询；

相关子查询中的每查询一条记录，需要重新做一次子查询；非相关子查询中子查询只需要进行一次；

相关子查询可以嵌套多层，但是嵌套层越多，效率越低；非相关子查询不可以嵌套，效率较高；

当子查询需要用到父查询的信息的时候就要用到相关自查询；

1. 实验总结

3.1 完成的工作

打开数据库

输入SQL语句

3.2 对实验的认识

掌握了复杂的查询，懂得了连接查询和子查询

3.3 遇到的困难及解决方法

无