

数据库系统课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 实验三 数据高级查询 |
| 实验日期： | 2023-4-21 |
| 实验地点： | 文宣楼B311 |
| 提交日期： | 2023-4-26 |
|  | |
| 学号： | 33920212204567 |
| 姓名： | 任宇 |
| 专业年级： | 软工2021级 |
| 学年学期： | 2022-2023学年第二学期 |

1. 实验环境

* 华为 ECS+openGauss 数据库服务器平台
* 前提：openGauss 数据库服务器正常运行
* 已建立带样例数据的 SALES 数据库

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

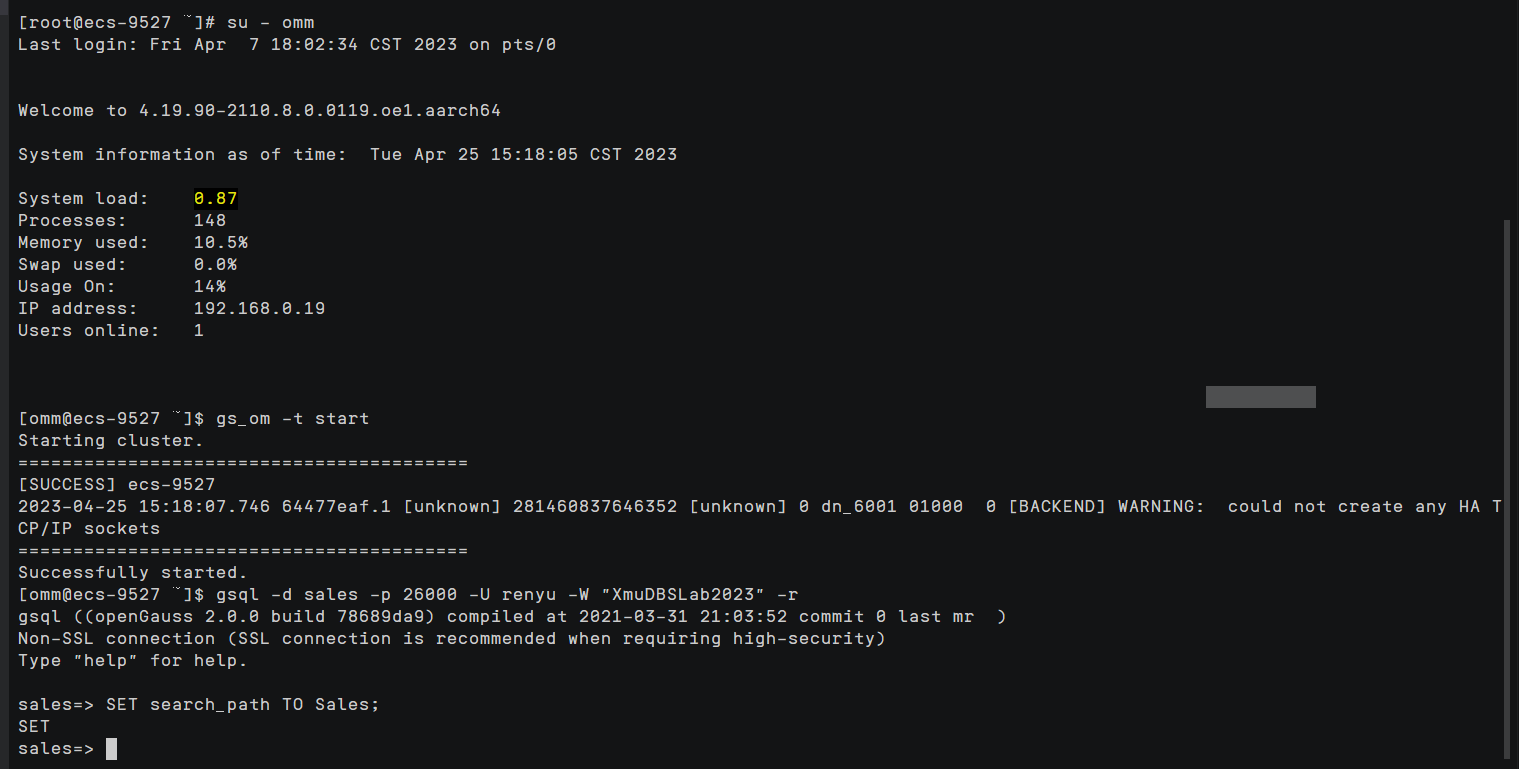
-ANY

-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掌握空值 NULL 的使用方法

1. 实验内容和步骤

（1）以 root 用户登录到 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sales：



（2）依次完成实验内容：

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

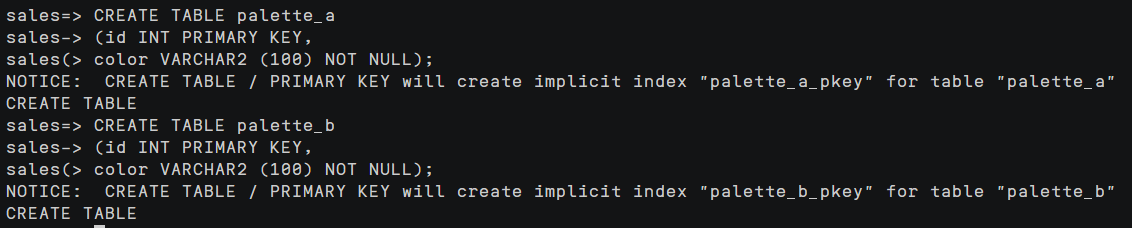
(id INT PRIMARY KEY,

color VARCHAR2 (100) NOT NULL);

CREATE TABLE palette\_b

(id INT PRIMARY KEY,

color VARCHAR2 (100) NOT NULL);



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

INSERT INTO palette\_a (id,color)

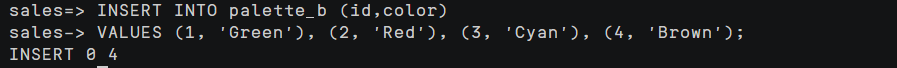
VALUES (1, 'Red'), (2, 'Green'), (3, 'Blue'), (4, 'Purple');



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

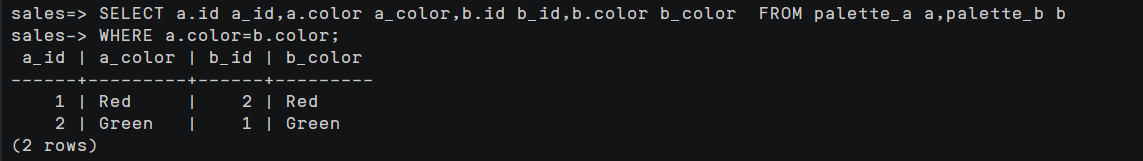
INSERT INTO palette\_b (id,color)

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 a.id a\_id,a.color a\_color,b.id b\_id,b.color b\_color FROM palette\_a a,palette\_b b WHERE a.color=b.color;

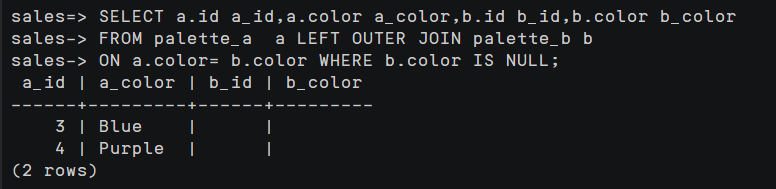


1. 查询 palette\_a 表中颜色不出现在 palette\_b 表中的 id 和颜色（左外连接）:

SELECT a.id a\_id,a.color a\_color,b.id b\_id,b.color b\_color

FROM palette\_a a LEFT OUTER JOIN palette\_b b

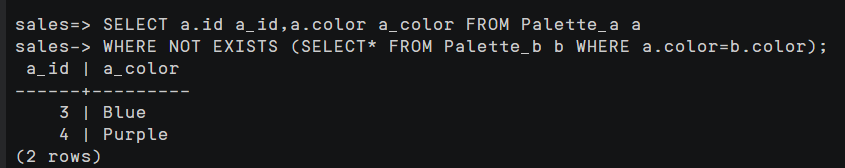
ON a.color= b.color WHERE b.color IS NULL;



使用子查询和NOT EXSITS：

SELECT a.id a\_id,a.color a\_color FROM Palette\_a a

WHERE NOT EXISTS (SELECT\* FROM Palette\_b b WHERE a.color=b.color);

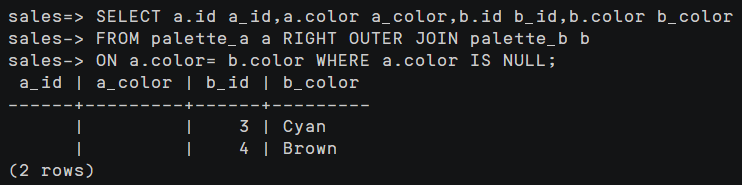


1. 查询 palette\_b 表中颜色不出现在 palette\_a 表中的 id 和颜色（右外连接）：

SELECT a.id a\_id,a.color a\_color,b.id b\_id,b.color b\_color

FROM palette\_a a RIGHT OUTER JOIN palette\_b b

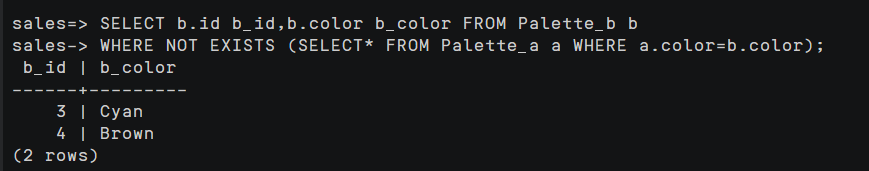
ON a.color= b.color WHERE a.color IS NULL;



使用子查询：

SELECT b.id b\_id,b.color b\_color FROM Palette\_b b

WHERE NOT EXISTS (SELECT\* FROM Palette\_a a WHERE a.color=b.color);

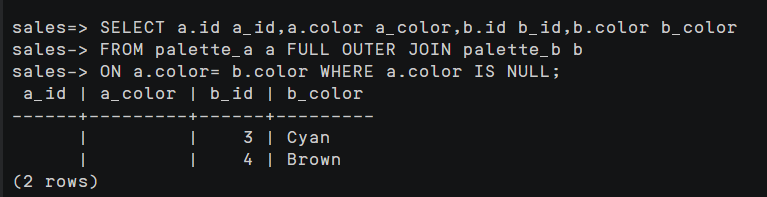


1. 查询（5）或（6）两种情况的信息（用（全）外连接）:

SELECT a.id a\_id,a.color a\_color,b.id b\_id,b.color b\_color

FROM palette\_a a FULL OUTER JOIN palette\_b b

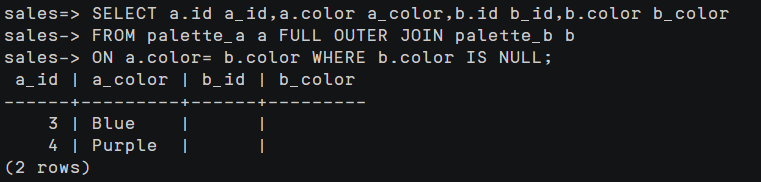
ON a.color= b.color WHERE b.color IS NULL;



SELECT a.id a\_id,a.color a\_color,b.id b\_id,b.color b\_color

FROM palette\_a a FULL OUTER JOIN palette\_b b

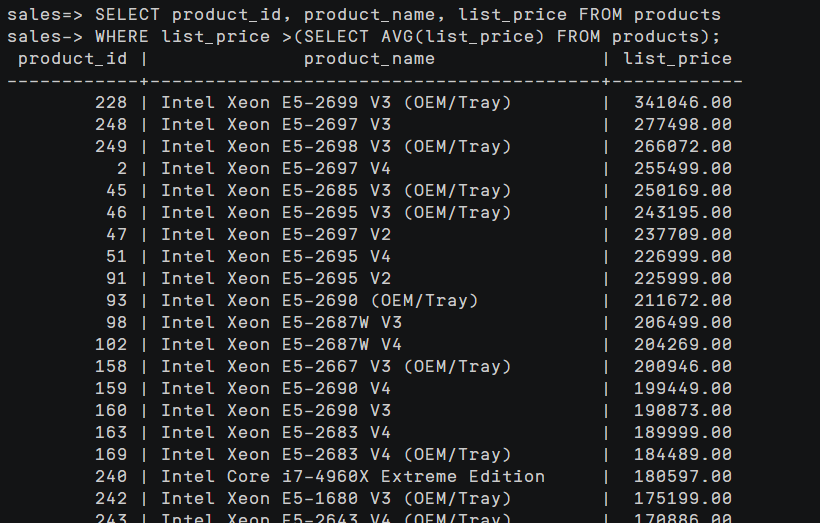
ON a.color= b.color WHERE a.color IS NULL;

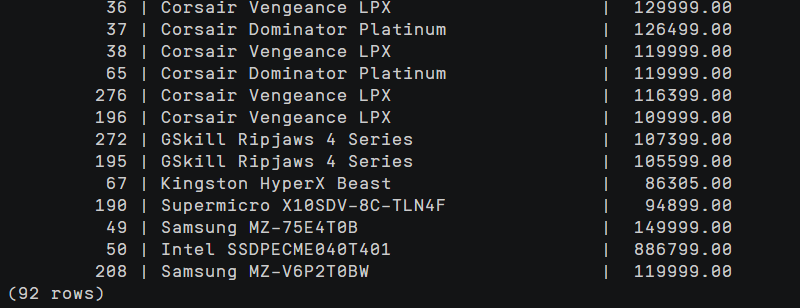


1. 查询产品表 products 中的 product\_id, product\_name, list\_price 信息，要求产品定价list\_price 大于其平均定价 list\_price:

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

WHERE list\_price >(SELECT AVG(list\_price) FROM products);



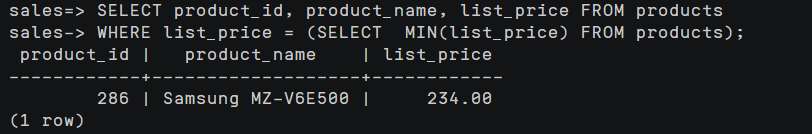


1. 查询产品表 products 中最便宜产品的 product\_id, product\_name, list\_price:

使用MIN聚集函数：

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

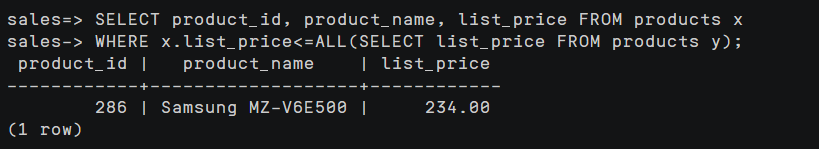
WHERE list\_price = (SELECT MIN(list\_price) FROM products);



使用ALL：

SELECT product\_id, product\_name, list\_price FROM products x

WHERE x.list\_price<=ALL(SELECT list\_price FROM products y);

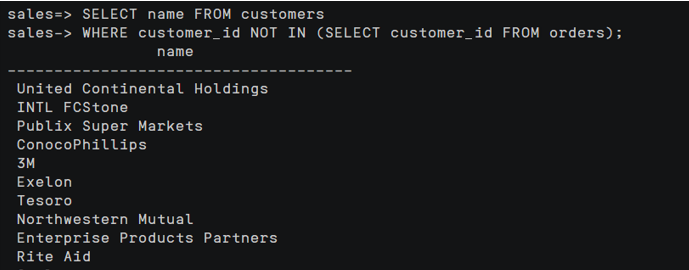


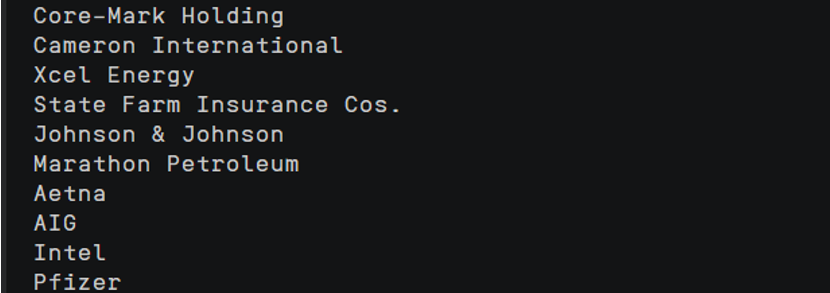
1. 查询没有一个订单的顾客姓名:

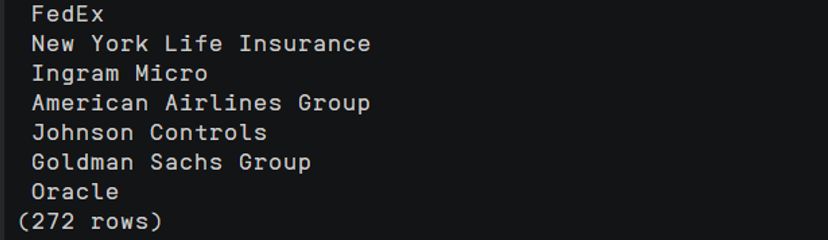
使用NOT IN：

SELECT name FROM customers

WHERE customer\_id NOT IN (SELECT customer\_id FROM orders);





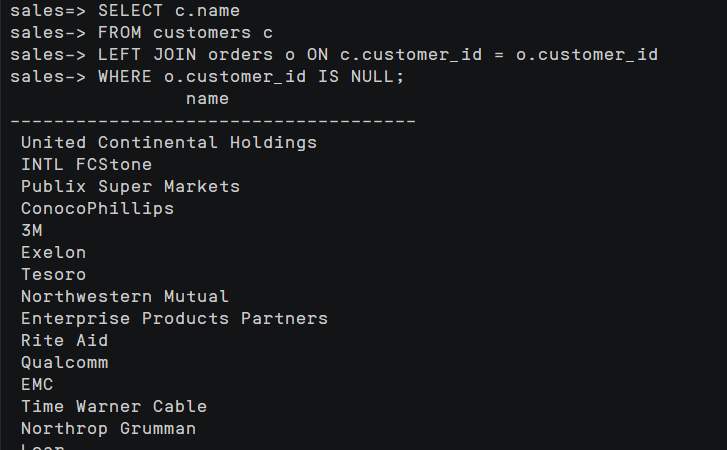


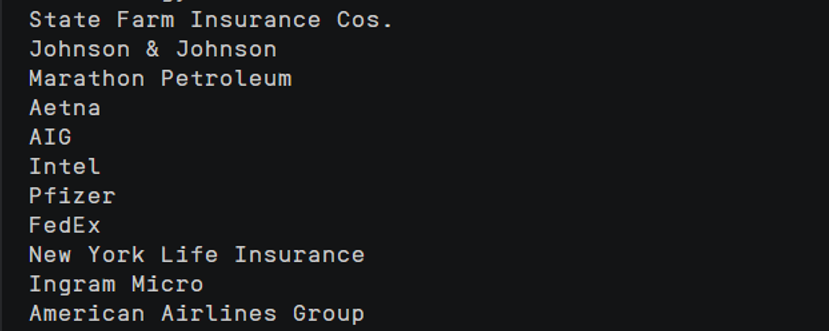
使用外连接：

SELECT c.name FROM customers c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.customer\_id IS NULL;





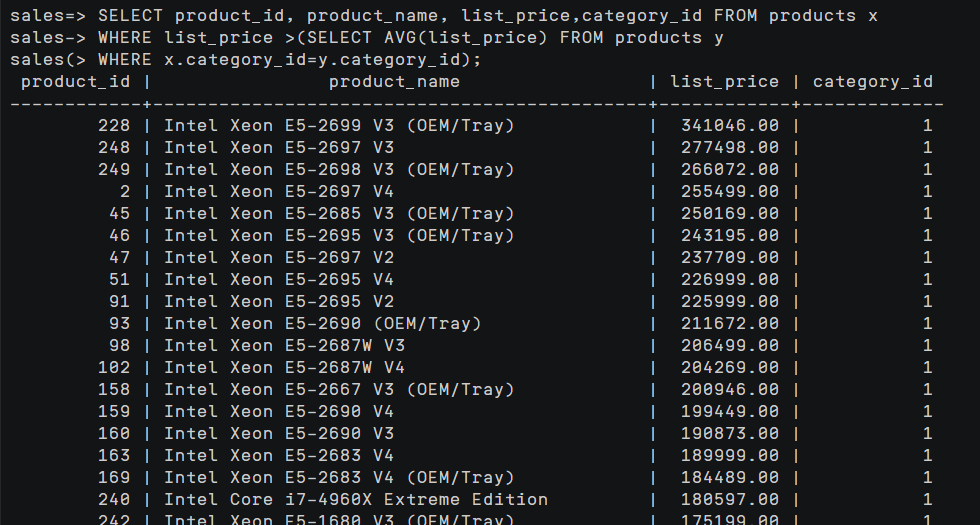
1. 查询产品表 products 中产品的 product\_id, product\_name, list\_price，要求产品定价list\_price 大于其同类产品（可由 category\_id 表达）的平均定价:

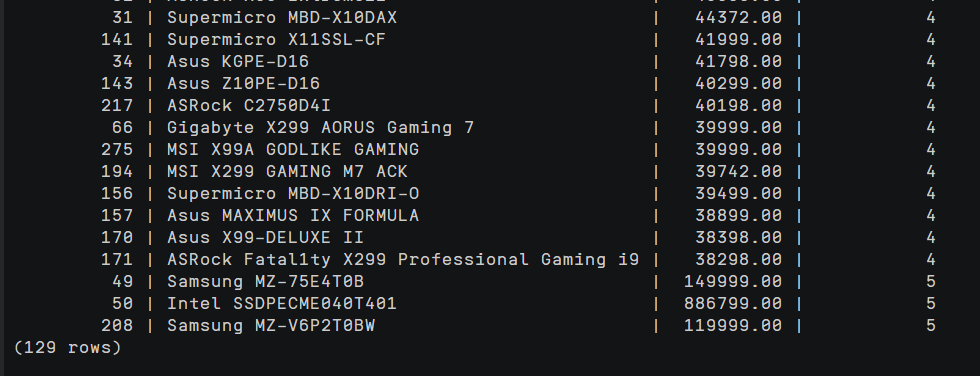
相关子查询：

SELECT product\_id, product\_name, list\_price,category\_id FROM products x

WHERE list\_price >(SELECT AVG(list\_price) FROM products y

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





内连接：

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

FROM products p

INNER JOIN (

SELECT category\_id, AVG(list\_price) AS avg\_price

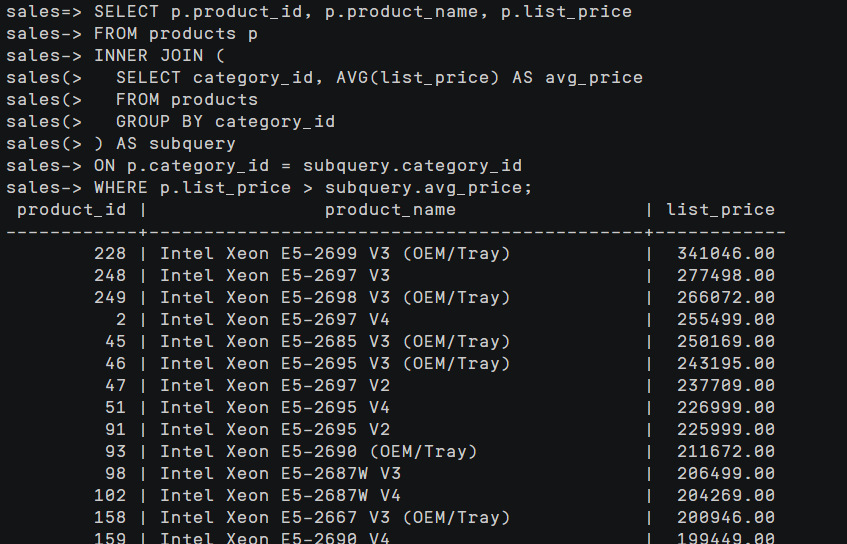
FROM products

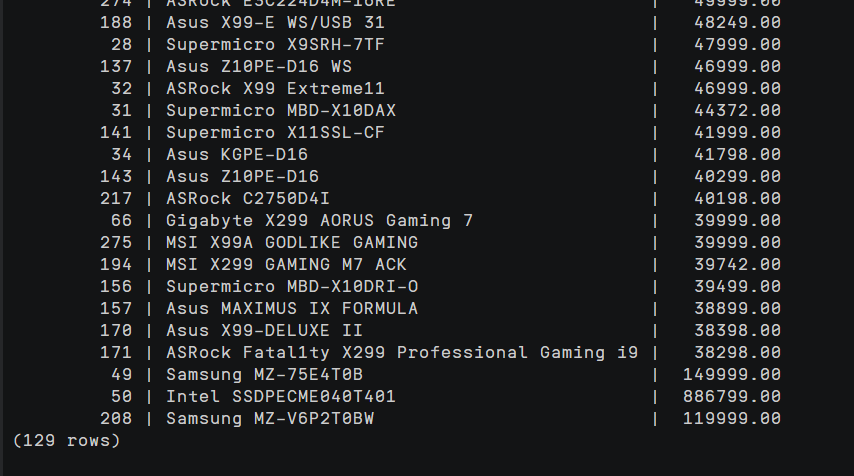
GROUP BY category\_id

) AS subquery

ON p.category\_id = subquery.category\_id

WHERE p.list\_price > subquery.avg\_price;





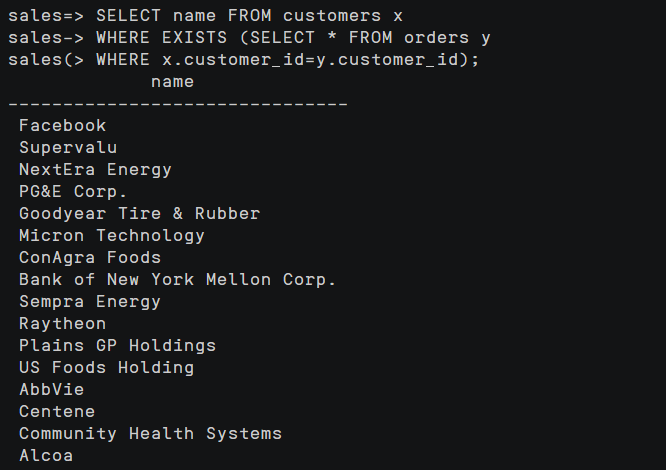
1. 查询有订单 order 的所有顾客 customer 姓名（查询涉及 customers 表和 orders 表）：

使用EXISTS：

SELECT name FROM customers x

WHERE EXISTS (SELECT \* FROM orders y

WHERE x.customer\_id=y.customer\_id);





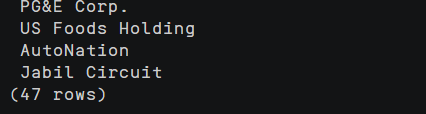
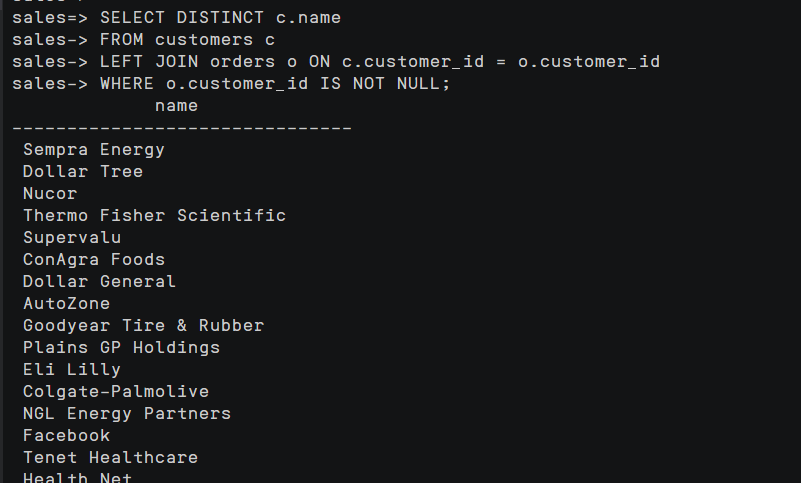
使用外连接：

SELECT DISTINCT c.name

FROM customers c

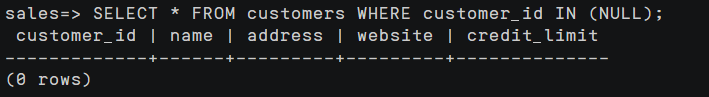
LEFT JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.customer\_id IS NOT NULL;

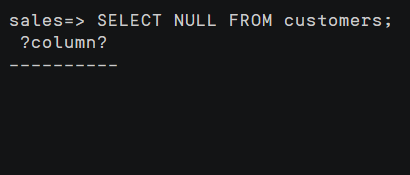


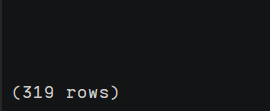
1. 执行以下三条与 NULL 相关的语句，观察各自执行的结果，能否从中得出某些初步结论？

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

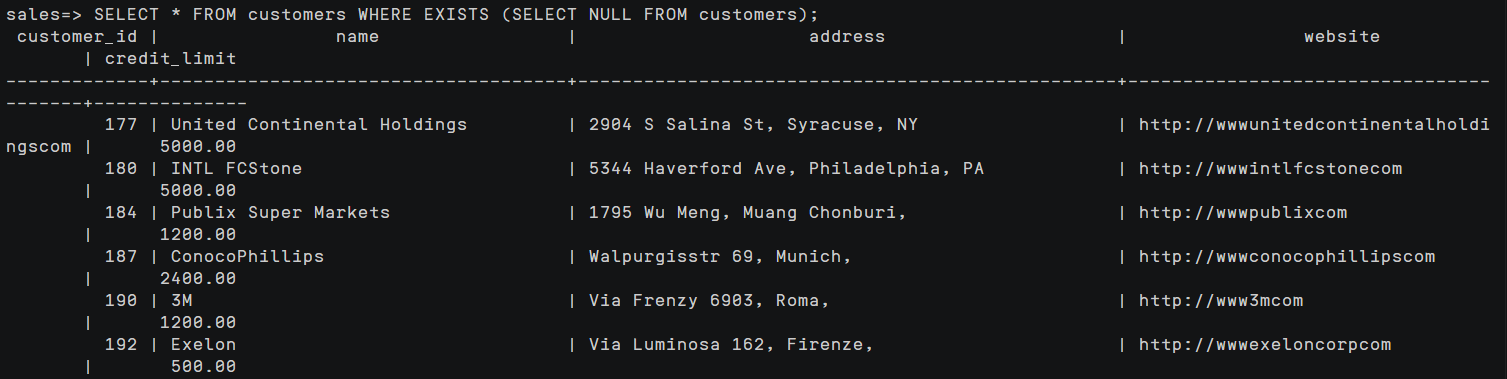


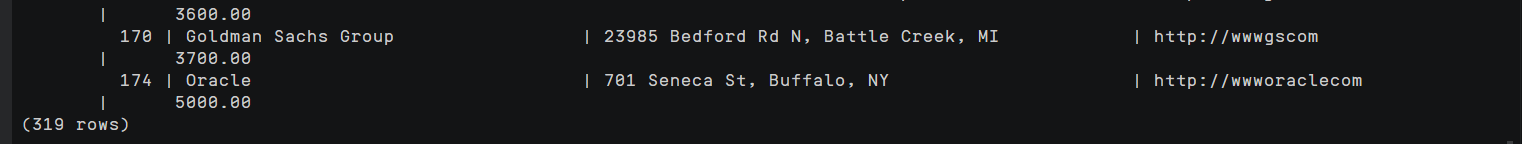
SELECT NULL FROM customers;





SELECT \* FROM customers WHERE EXISTS (SELECT NULL FROM customers);





第一条命令查询结果0条，第二、三条查询结果都是319行，但第二条没有内容，第三有内容完整。

结论：当NULL作为查询条件时，空集的子集还是空集，没有任何元组属于空集，故查询结果0条。

当NULL作为查询项时，尽管查询项内容为空，但还是会显示所有项，即319条无内容的元组。

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

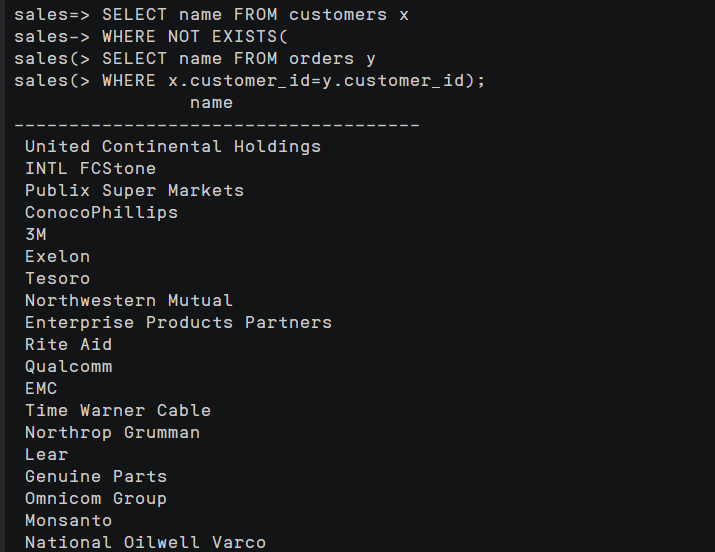
使用NOT EXISTS：

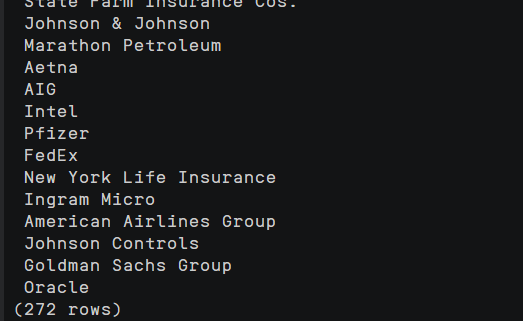
SELECT name FROM customers x

WHERE NOT EXISTS(

SELECT name FROM orders y

WHERE x.customer\_id=y.customer\_id);





使用外连接：

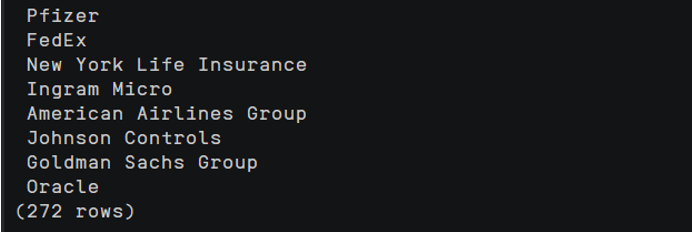
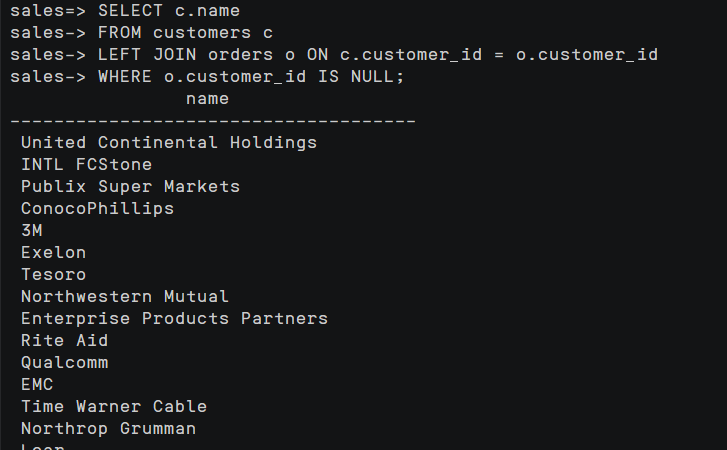
SELECT DISTINCT x.name

FROM customers x

LEFT JOIN orders y

ON x.customer\_id = y.customer\_id

WHERE y.customer\_id IS NULL;



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:

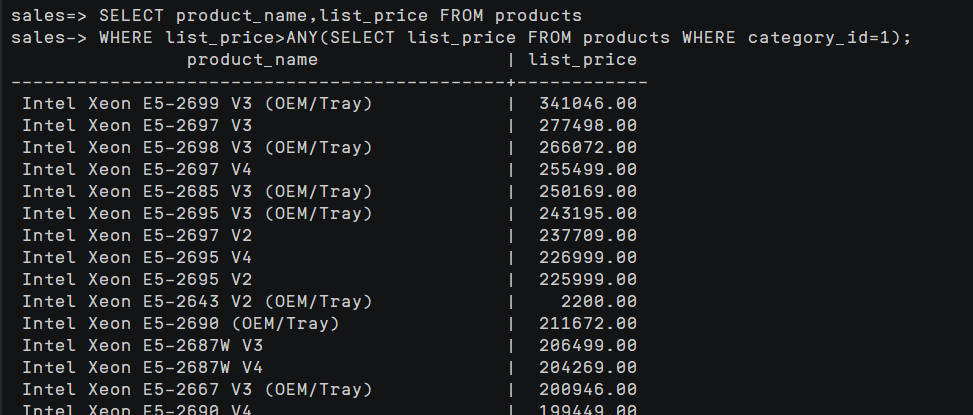
使用ANY：

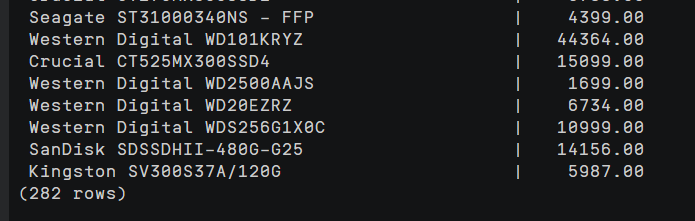
SELECT product\_name,list\_price FROM products

WHERE list\_price>ANY (SELECT list\_price

FROM products

WHERE category\_id=1);





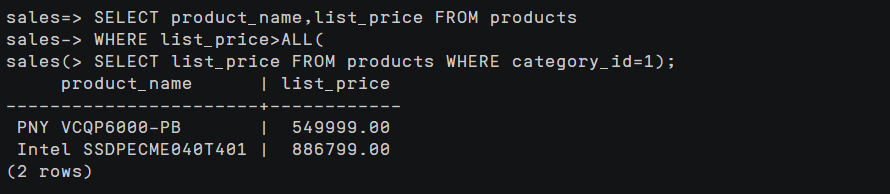
1. 查询产品表 products 中的产品名 product\_name 和定价 list\_price，要求其定价高于产品种类 1 中的所有定价:

使用ALL：

SELECT product\_name,list\_price FROM products

WHERE list\_price>ALL(

SELECT list\_price FROM products WHERE category\_id=1);



使用MAX聚集函数：

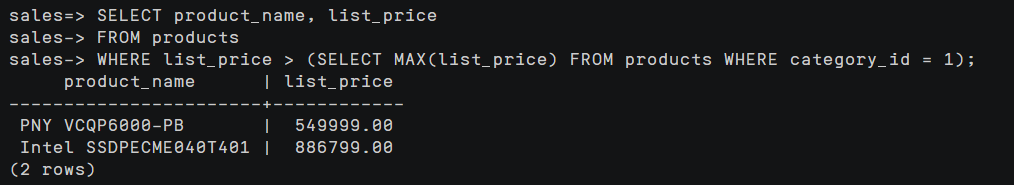
SELECT product\_name, list\_price

FROM products

WHERE list\_price > (SELECT MAX(list\_price)

FROM products

WHERE category\_id = 1);



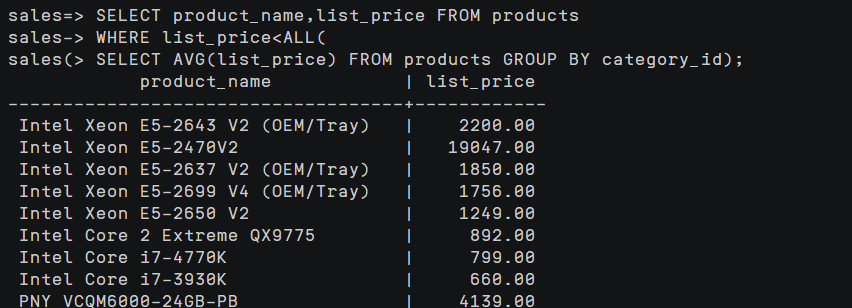
1. 查询产品表 products 中的产品名 product\_name 和定价 list\_price，要求其定价低于产品种类的所有平均定价:

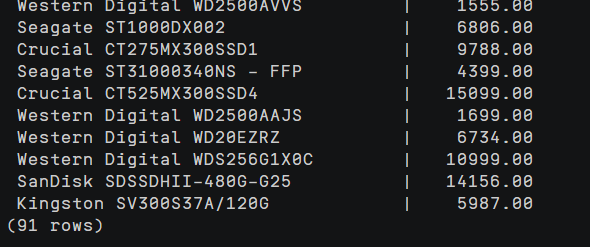
使用ALL：

SELECT product\_name,list\_price FROM products

WHERE list\_price<ALL(

SELECT AVG(list\_price) FROM products GROUP BY category\_id);





使用MIN聚集函数：

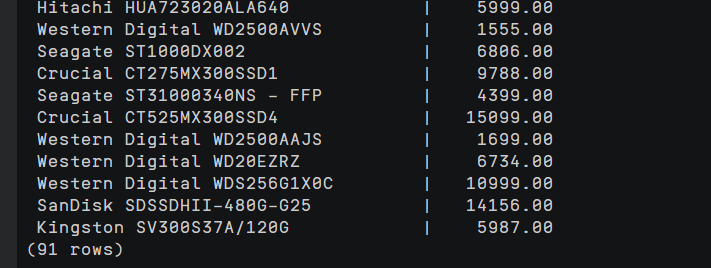
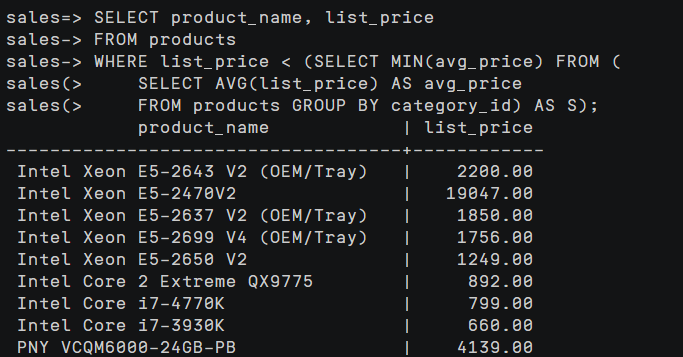
SELECT product\_name, list\_price

FROM products

WHERE list\_price < (SELECT MIN(avg\_price) FROM (

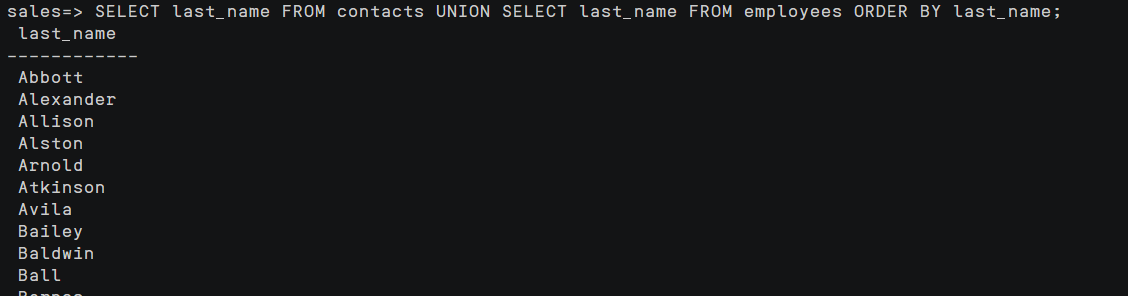
SELECT AVG(list\_price) AS avg\_price

FROM products GROUP BY category\_id) AS S;



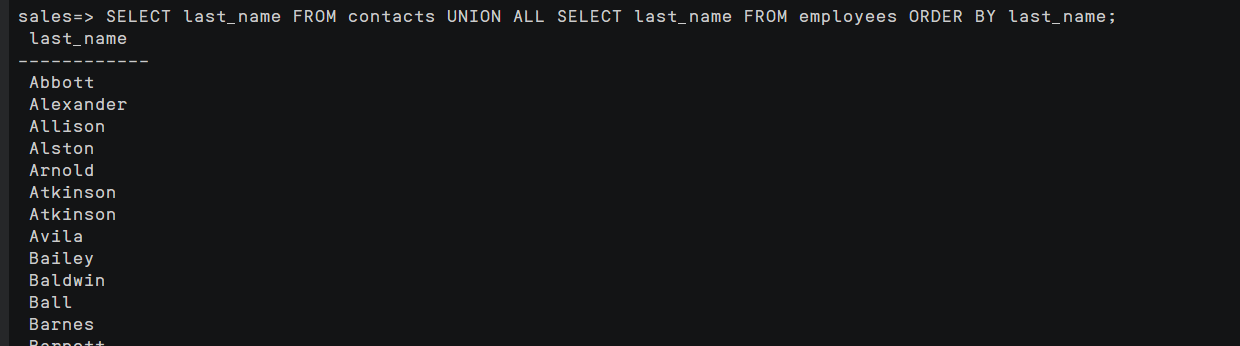
1. 查询 contacts 表和 employees 表中的所有 last\_name，并以 last\_name 升序显示:

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



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 contacts UNION ALL SELECT last\_name FROM employees ORDER BY last\_name;

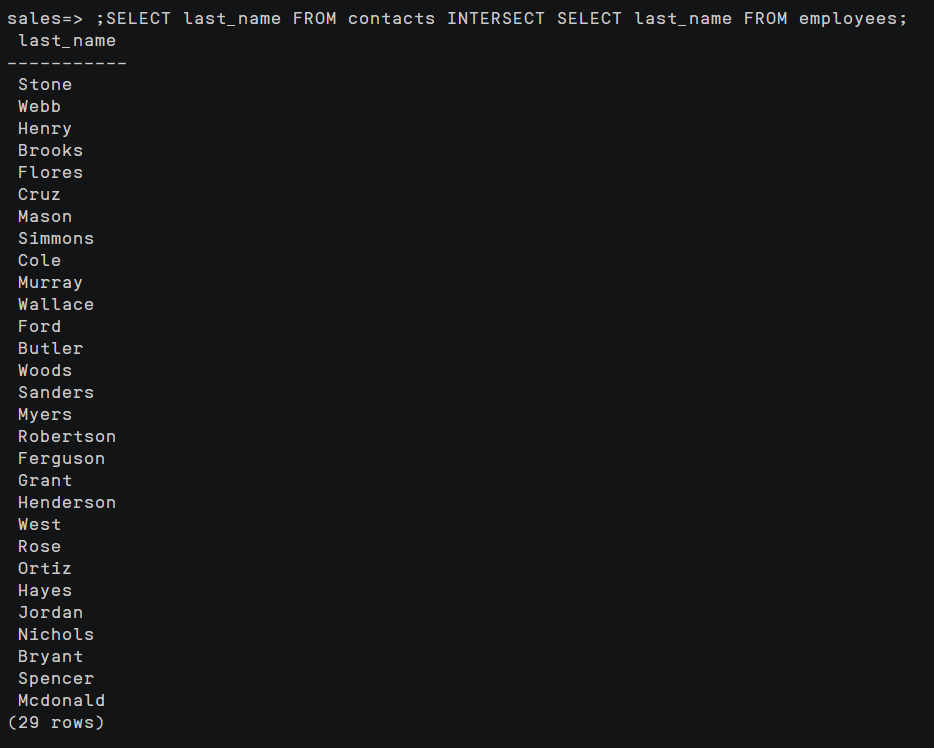




1. 查询同时出现在 contacts 表和 employees 表中的所有 last\_name:

使用INTERSECT：

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



使用普通查询：

SELECT DISTINCT x.last\_name FROM contacts x,employees y where x.last\_name=y.last\_name;

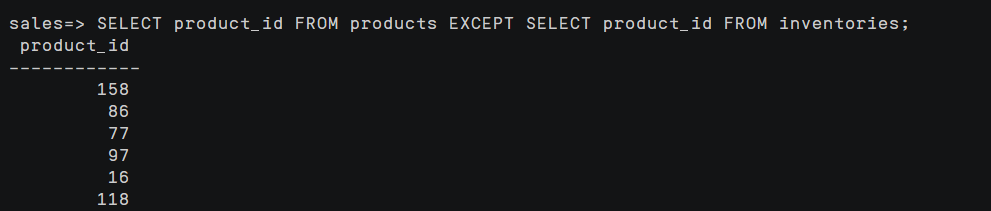


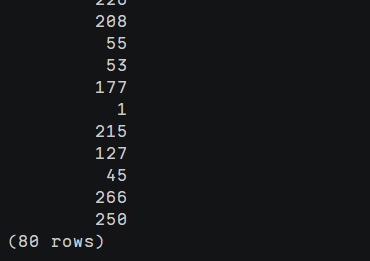


1. 查询在产品表 products 中而不在库存表 inventories 中的产品号 product\_id:

使用EXCEPT：

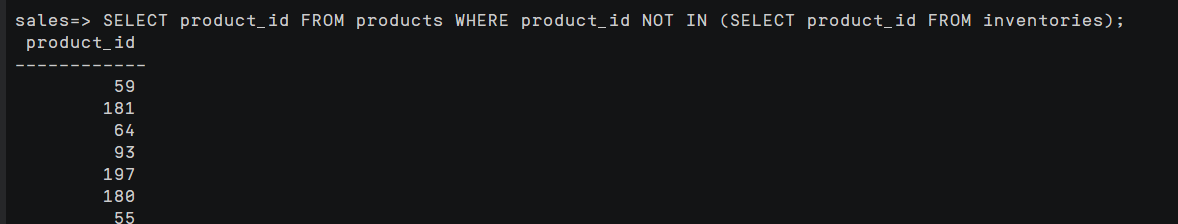
SELECT product\_id FROM products EXCEPT SELECT product\_id FROM inventories;

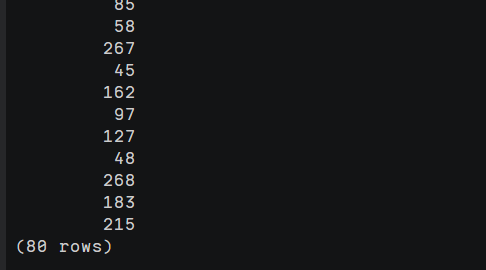




其他查询方法：

SELECT product\_id FROM products WHERE product\_id NOT IN (SELECT product\_id FROM inventories);





1. 实验总结

4.1 完成的工作

1. 登录到创建的Sales数据库中；

2. 使用SQL语句进行查询操作，完成了实验内容；

4.2 对实验的认识

通过本次实验，练习了SQL语言中比较复杂的查询语句，同时也学习了左连接，右连接，全外连接的区别，对使用SQL语法进行子查询和派生表查询有了更深的理解和掌握。

4.3 遇到的困难及解决方法

碰到的困难是如何使用其他查询语句替代UNION的操作，但还未找到较为合适的方法，会继续研究。

1. 实验思考

 （1）什么类型的查询只能用子查询实现？试举例说明。

不管是否在同一个表，当表业务逻辑出现分级，就需要用子查询，比如说查所有学过计算机网络的同学的数据结构成绩。

 （2）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）

相关子查询和不相关子查询的区别在于，相关子查询需要依赖于外部查询语句中的字段或者表，而不相关子查询不需要依赖于外部查询。

相关子查询应用场景：

* 子查询中需要用到外部查询的某些列进行计算的情况；
* 子查询需要用到外部查询的某些列作为过滤条件的情况；

要将相关子查询转换为一般查询，可以采取以下两种方法：

* 使用JOIN替代子查询：可以通过使用LEFT JOIN、INNER JOIN等关联查询，将子查询所需的字段从子查询提前到JOIN操作中，并且使用ON子句来指定连接条件。
* 使用临时表替代子查询：可以先将子查询的结果保存到一个临时表中，然后再在外部查询中引用该临时表。