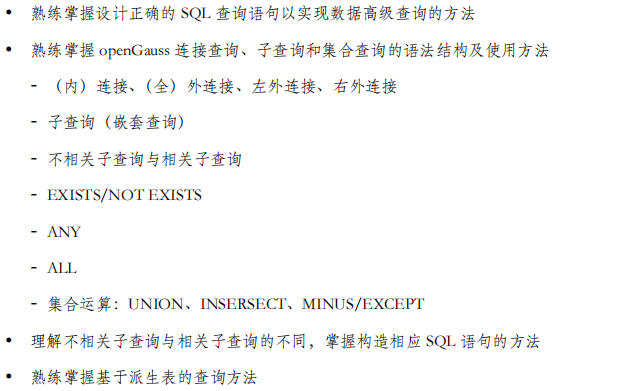


数据库系统课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 数据高级查询 |
| 实验日期： | 2022/4/7 |
| 实验地点： | 4号楼 |
| 提交日期： | 2022/4/19 |
|  | |
| 学号： | 22920202202879 |
| 姓名： | 陈奕培 |
| 专业年级： | 软工2020级 |
| 学年学期： | 2021-2022学年第二学期 |

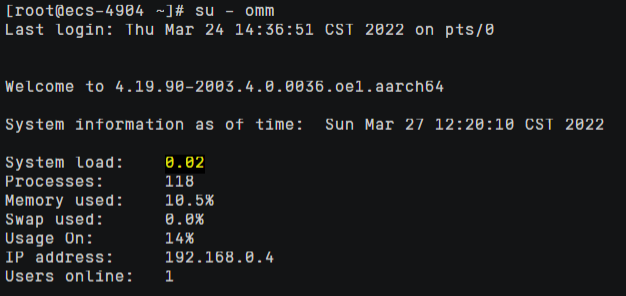
1. 实验目的



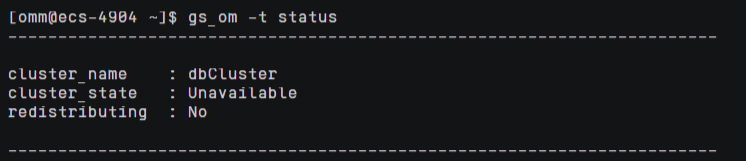
1. 实验内容和步骤

（1）连接到数据库

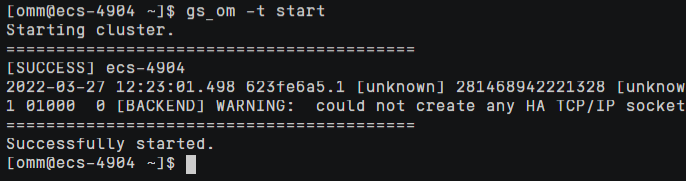
1.在数据库主节点服务器上，切换至omm操作系统用户环境。



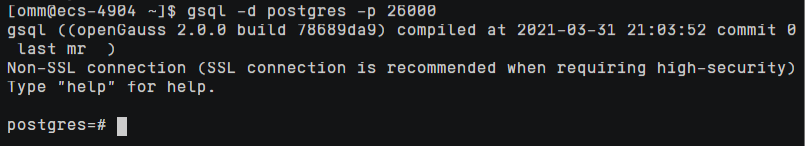
2.查看服务是否启动。



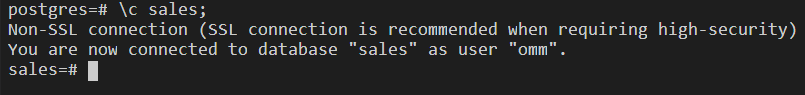
3.启动服务。



4.连接到数据库。



5.进入sales数据库



（2）查询

1. 创建两张表

CREATE TABLE palette\_a(

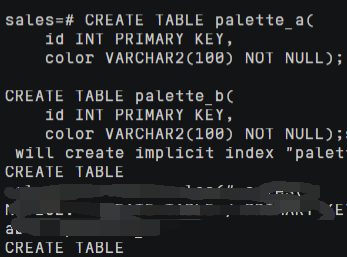
    id INT PRIMARY KEY,

    color VARCHAR2(100) NOT NULL);

CREATE TABLE palette\_b(

    id INT PRIMARY KEY,

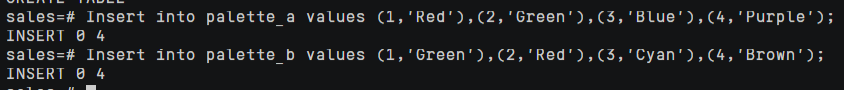
    color VARCHAR2(100) NOT NULL);



2.3.插入数据

Insert into palette\_a values (1,'Red'),(2,'Green'),(3,'Blue'),(4,'Purple');

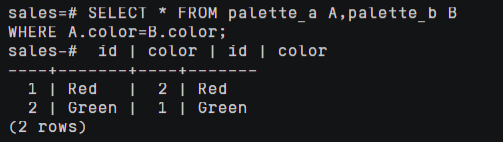
Insert into palette\_b values (1,'Green'),(2,'Red'),(3,'Cyan'),(4,'Brown');



（4）查询两张表中相同颜色的所有信息。

SELECT \* FROM palette\_a A,palette\_b B

WHERE A.color=B.color;



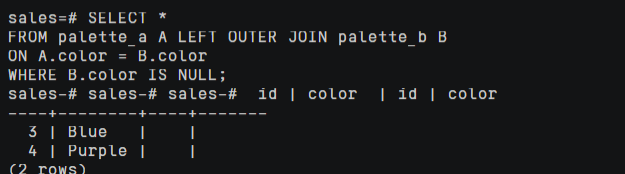
（5）查询 palette\_a 表中颜色不出现在 palette\_b 表中的两张表的 id 和颜色（用左外连接）。

SELECT \*

FROM palette\_a A LEFT OUTER JOIN palette\_b B

ON A.color = B.color

WHERE B.color IS NULL;



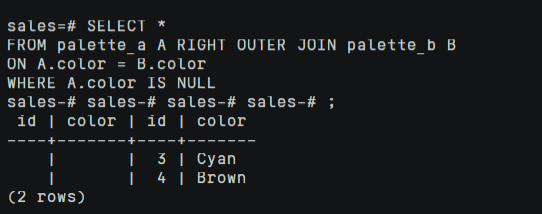
（6）查询 palette\_b 表中颜色不出现在 palette\_a 表中的两张表的 id 和颜色（用右外连接）。

SELECT \*

FROM palette\_a A RIGHT OUTER JOIN palette\_b B

ON A.color = B.color

WHERE A.color IS NULL;



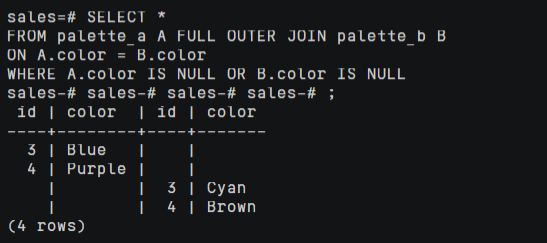
（7）查询（5）或（6）两种情况的信息（用（全）外连接）。

SELECT \*

FROM palette\_a A FULL OUTER JOIN palette\_b B

ON A.color = B.color

WHERE A.color IS NULL OR B.color IS NULL;



（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

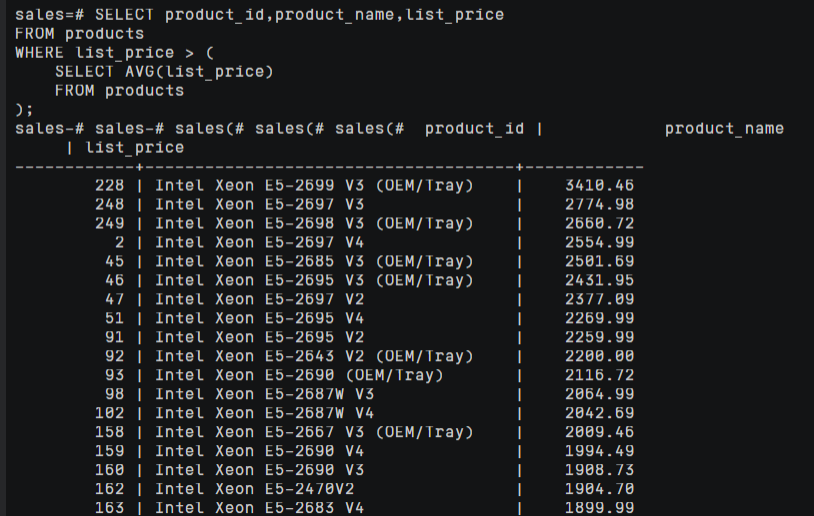
FROM products

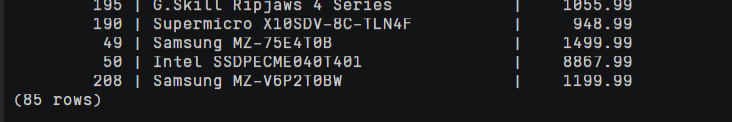
WHERE list\_price > (

SELECT AVG(list\_price)

FROM products

);





（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

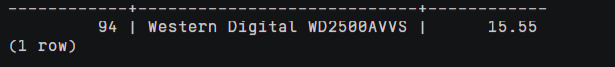
FROM products

WHERE list\_price = (

SELECT MIN(list\_price)

FROM products

);



（10）查询没有一个订单的顾客姓名。

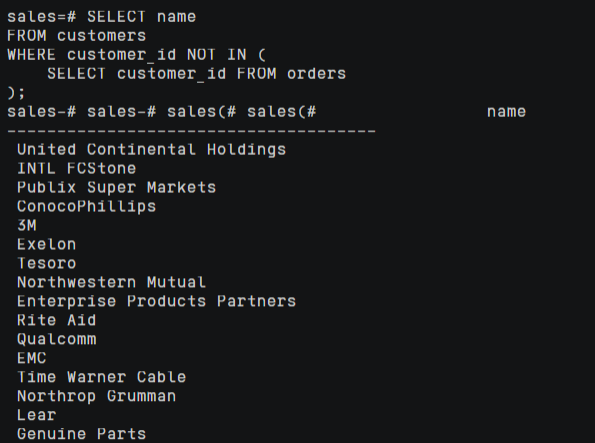
SELECT name

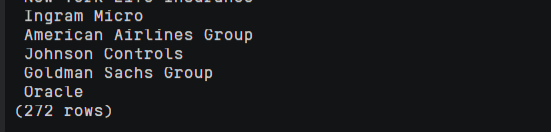
FROM customers

WHERE customer\_id NOT IN (

SELECT customer\_id FROM orders

);





（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。

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

FROM products A

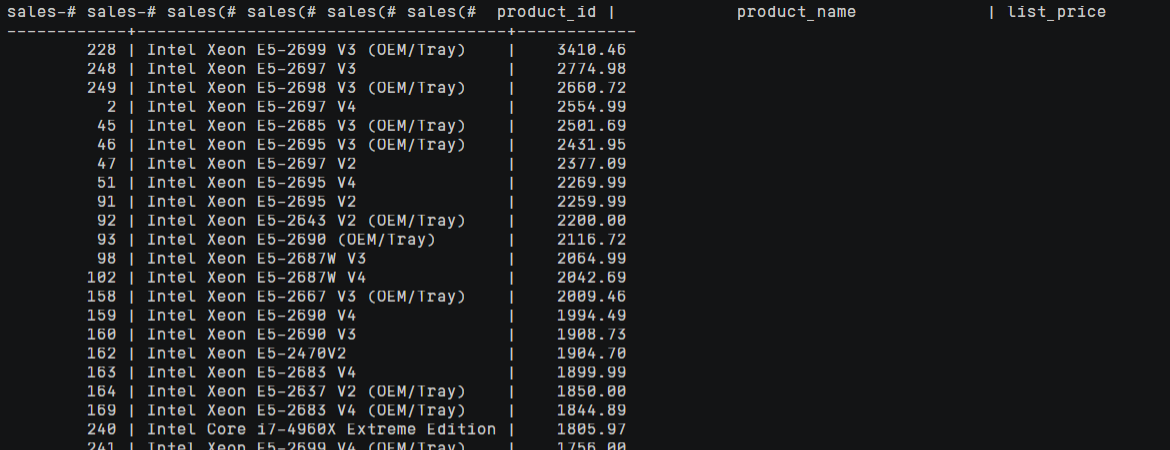
WHERE list\_price > (

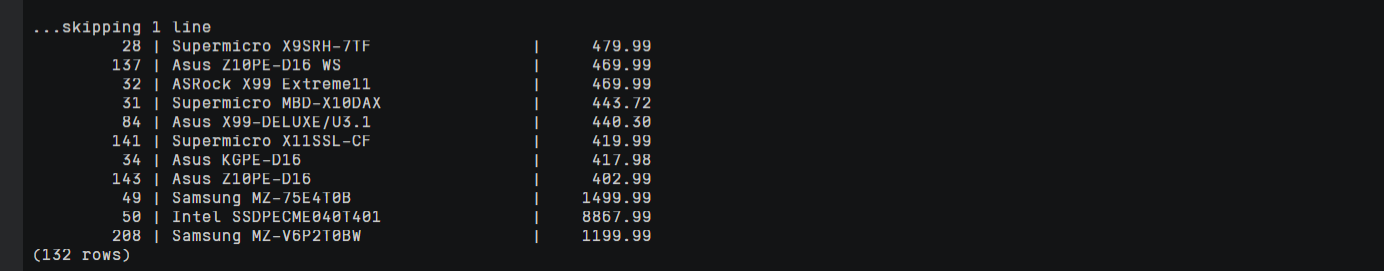
SELECT AVG(list\_price)

FROM products B

WHERE A.category\_id = B.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）。

SELECT name

FROM customers

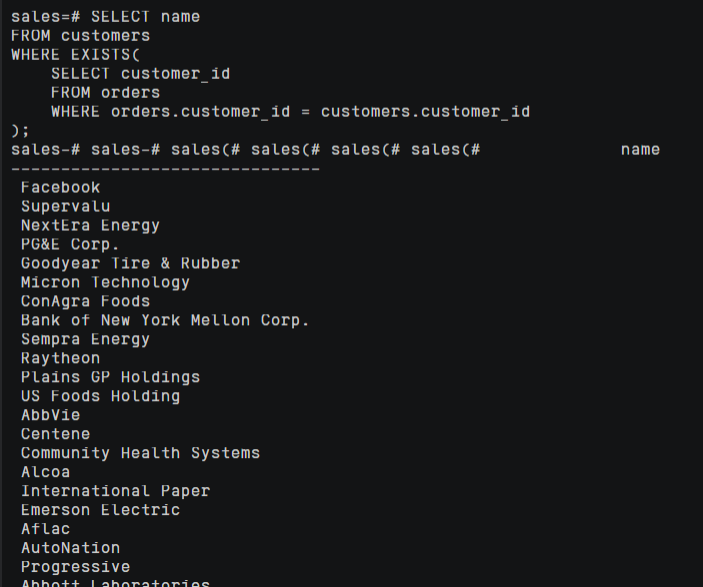
WHERE EXISTS(

SELECT customer\_id

FROM orders

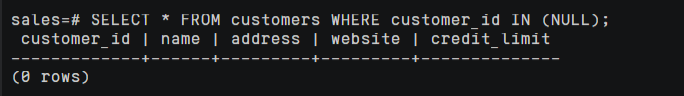
WHERE orders.customer\_id = customers.customer\_id

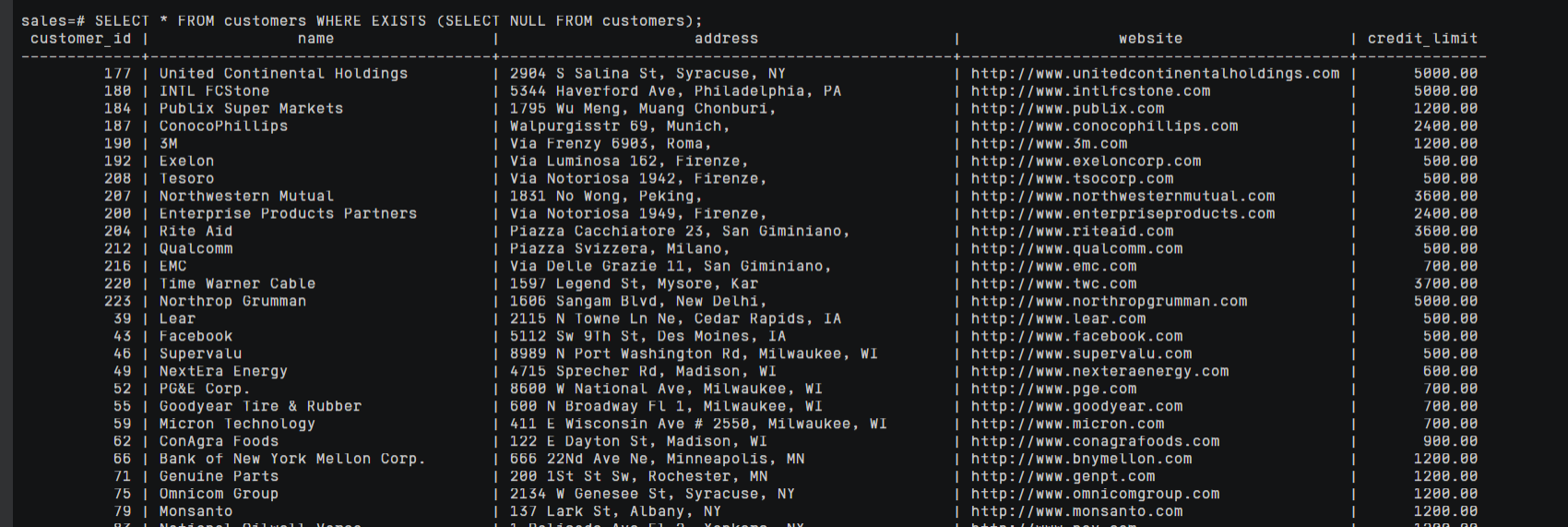
);





13.





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，证明SELECT NULL FROM table返回值为1；

（14）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 customers

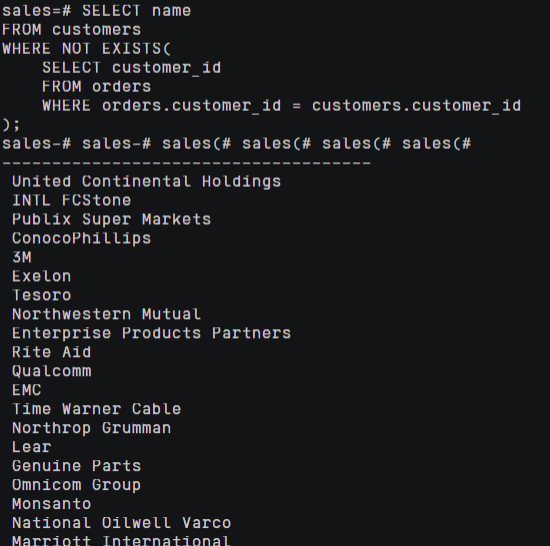
WHERE NOT EXISTS(

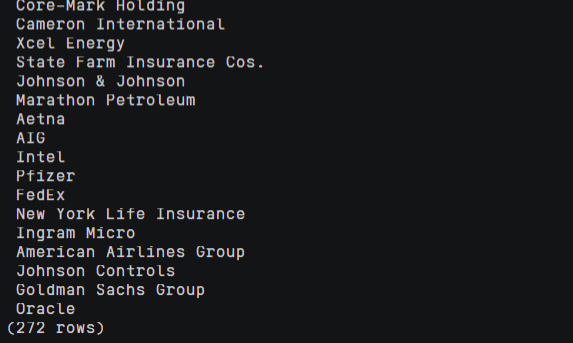
SELECT customer\_id

FROM orders

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

);





（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。

SELECT product\_name,list\_price

FROM products

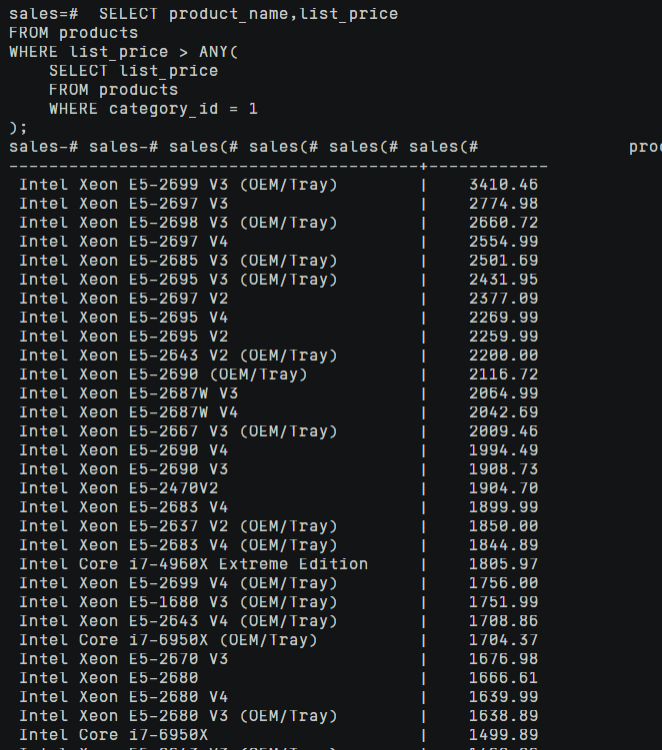
WHERE list\_price > ANY(

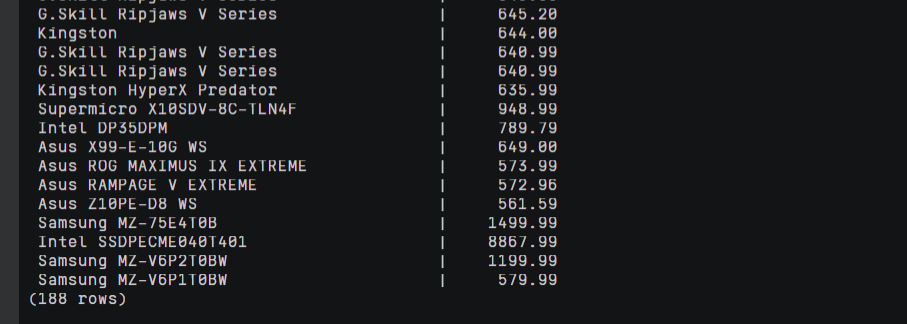
SELECT list\_price

FROM products

WHERE category\_id = 1

);





（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 products

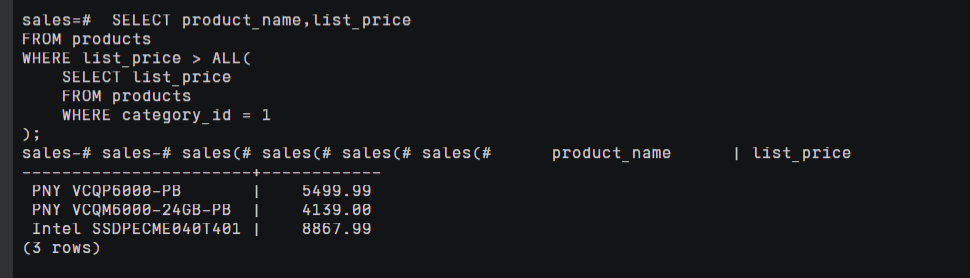
WHERE list\_price > ALL(

SELECT list\_price

FROM products

WHERE category\_id = 1

);



（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。

SELECT product\_name,list\_price

FROM products

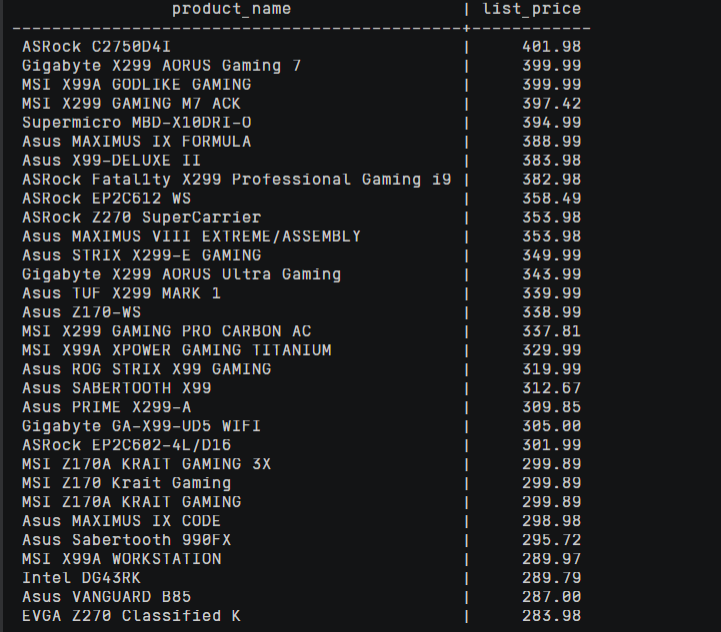
WHERE list\_price < ALL(

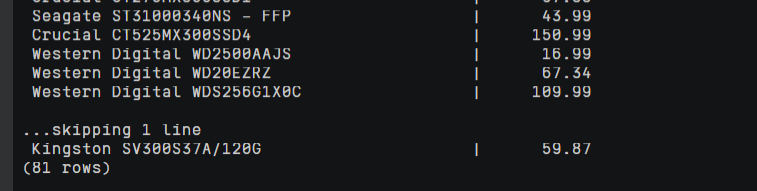
SELECT AVG(list\_price)

FROM products

GROUP BY category\_id

);





（18）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(

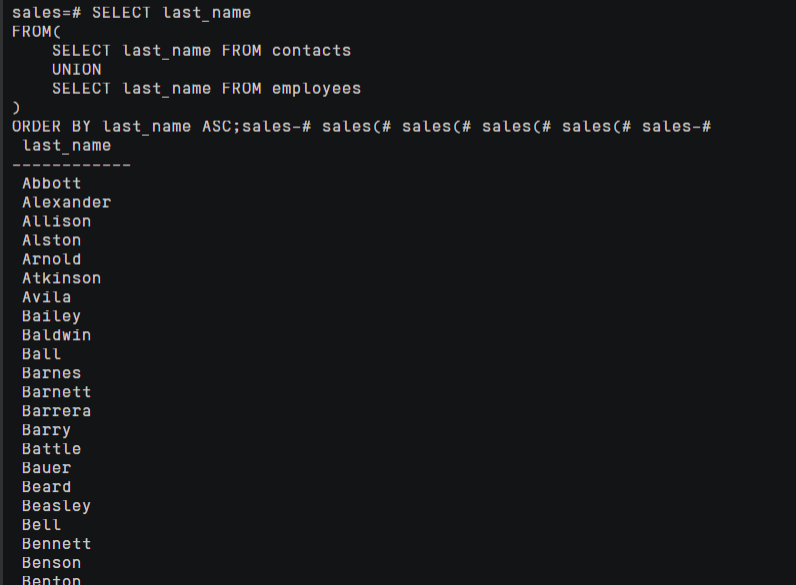
SELECT last\_name FROM contacts

UNION

SELECT last\_name FROM employees

)

ORDER BY last\_name ASC;





（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(

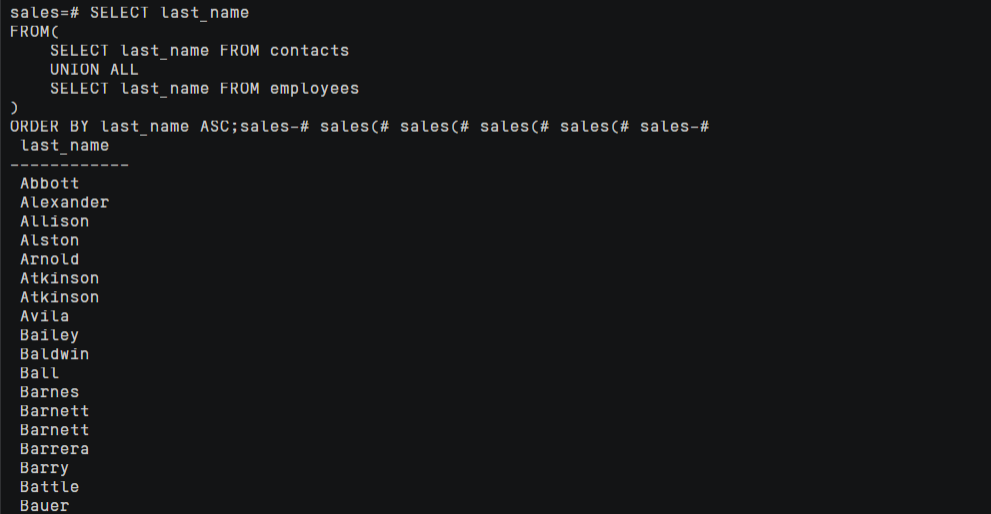
SELECT last\_name FROM contacts

UNION ALL

SELECT last\_name FROM employees

)

ORDER BY last\_name ASC;





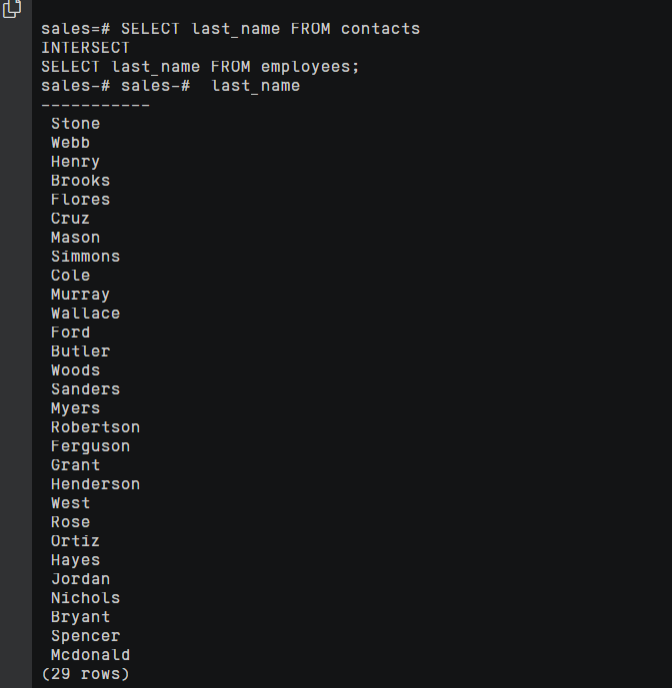
（20）查询同时出现在 contacts 表和 employees 表中的所有 last\_name。

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;



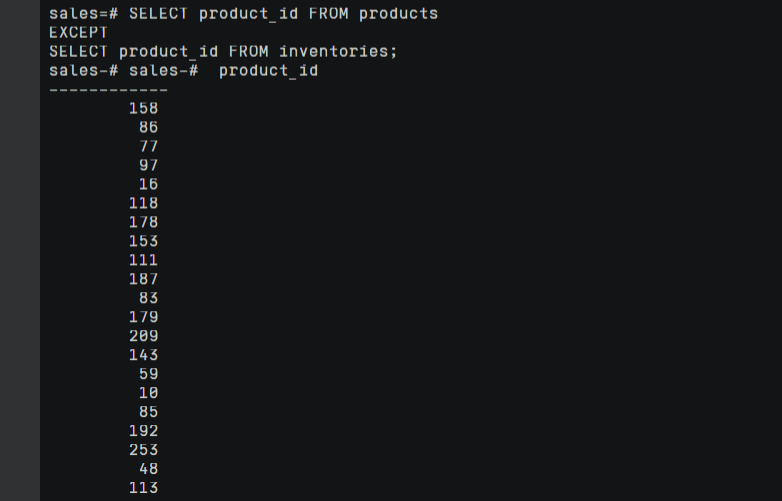
（21）查询在产品表 products 中而不在库存表 inventories 中的产品号 product\_id。

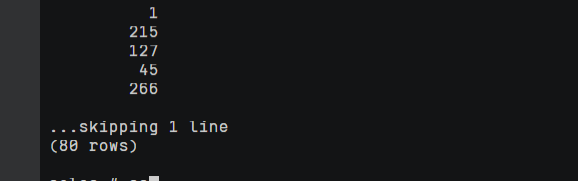
实现要求：MINUS/EXCEPT（必须）+其它查询方法（如果找到）

SELECT product\_id FROM products

EXCEPT

SELECT product\_id FROM inventories;





（3）思考：

* 当查询所需要的数据不能直接获得时，就需要使用子查询；如查询没有一个订单的顾客姓名；
* 区别：1. 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；2. 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；3. 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；
* 基于派生表的查询可以转换成一般查询；

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