**数据库实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实 验 人：** | **凌国明** | | **学 号：** | **21307077** | **日 期：** | **2023年9月7日** |
| **院（系）：** | **计算机学院** | | | **专业（班级）：** | **计算机科学与技术** | |
| **联系方式：** |  | | |  |  | |
| **实验题目：** | | **1.2 数据基本查询** | | | | |

* + - 1. **实验目的**

掌握 SQL 程序设计基本规范，熟练运用 SQL 语言实现数据基本查询，包括单表查询、 分组统计查询和连接查询。

* + - 1. **实验内容和要求**

针对 TPC-H 数据库设计各种单表查询 SQL 语句、分组统计查询语句；设计单个表对自身的连接查询，设计多个表的连接查询。理解和掌握 SQL 查询语句各个子句的特点和作用，按照 SQL 程序设计规范写出具体的 SQL 查询语句，并调试通过。

说明：简单地说，SQL 程序设计规范包含 SQL 关键字大写、表名、属性名、存储过程名等标识符大小写混合、SQL 程序书写缩进排列等编程规范。

* + - 1. **实验重点和难点**

实验重点：分组统计查询、单表自身连接查询、多表连接查询。

实验难点：区分元组过滤条件和分组过滤条件；确定连接属性，正确设计连接条件。

* + - 1. **实验工具**

MySQL、SQL Server、Navicat

* + - 1. **实验过程**

（1）单表查询（实现投影操作）

查询供应商的名称、地址和联系电话。

SELECT name,address,phone

FROM Supplier;

部分结果截屏如下：



（2）单表查询（实现选择操作）

查询在'2014-05-14' 和 '2014-05-15'这两天内提交的总价大于 1000 元的订单的编号、顾客编号等订单的所有信息。

* 若没有计算订单明细的总价（公式是：**extendedprice = quantity \* Part.retailprice**），则执行以下语句：

UPDATE Lineitem

SET extendedprice = quantity \* (select retailprice from part where partkey=Lineitem.partkey);

* 若**Lineitem.tax** is NULL，则执行以下语句：

UPDATE Lineitem

SET tax=0

WHERE tax IS NULL;

* 若没有计算订单的总价（公式是：**totalprice= SUM(Lineitem.extendedprice\*(1-Lineitem.discount)\*(1+Lineitem.tax))**），则执行以下语句：

UPDATE Orders

SET totalprice=ROUND((select sum(li.extendedprice\*(1-li.discount)\*(1+li.tax)) from lineitem li where li.orderkey=Orders.orderkey),2);

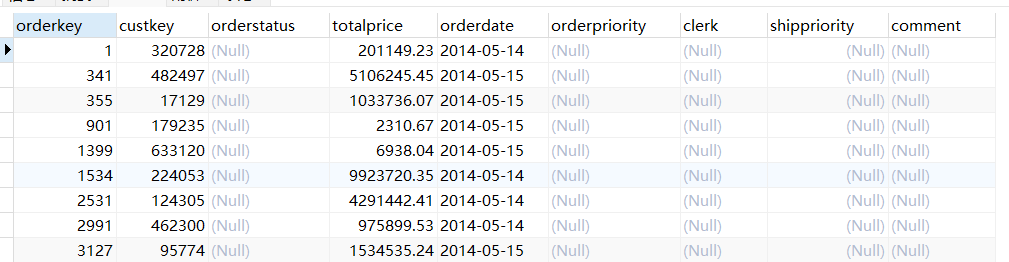
再执行以下语句：

SELECT \*

FROM orders

WHERE totalprice>1000 AND (orderdate='2014-05-14' OR orderdate='2014-05-15')

部分结果截屏如下：



（3）不带分组过滤条件的分组统计查询

统计每个顾客的订购金额。

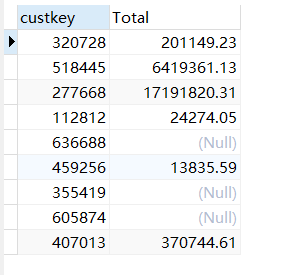
执行以下SQL语句：

SELECT custkey, SUM(totalprice) as Total

FROM orders

GROUP BY custkey;

部分结果截屏如下：



（4）带分组过滤条件的分组统计查询

查询其订单数目至少有2个订单并且其订单平均金额超过 1000元的顾客编号、姓名、订单数和订单平均金额。

执行以下SQL语句：

SELECT customer.custkey, customer.name, COUNT(orderkey) AS order\_num, AVG(totalprice) AS avg\_price

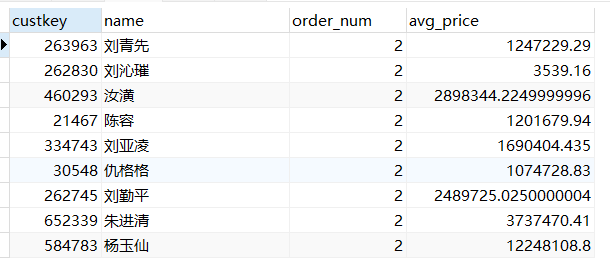
FROM orders, customer

WHERE orders.custkey=customer.custkey

GROUP BY orders.custkey

HAVING COUNT(orderkey)>=2 AND AVG(totalprice)>1000;

部分结果截屏如下：



（5）单表自身查询

查询与“上海黎顺服装经营部”在同一个国家的供应商编号、名称和地址信息。

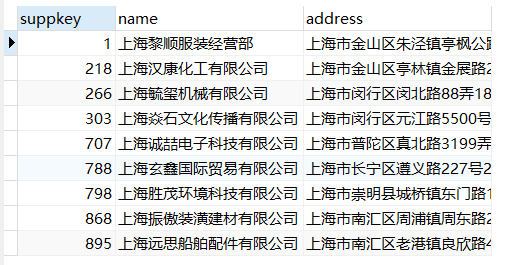
执行以下SQL语句：

SELECT suppkey, name, address

FROM supplier

WHERE nationkey=(SELECT nationkey FROM supplier WHERE name='上海黎顺服装经营部')

部分结果截屏如下：



（6）两表连接查询（普通连接）

计算Part和partsupp的笛卡尔乘积，从中找出其零售价格为10、而供应价格为0的元组中的零件名、制造商名、零售价格和供应价格。

执行以下SQL语句：

SELECT name, mfgr, retailprice, supplycost

FROM part, partsupp

WHERE retailprice=10 AND supplycost=0

部分结果截屏如下：



（7）两表连接查询（自然连接）

查询其零售价格大于其某个供应价格的10000倍的零件名、制造商名、零售价格和供应价格。

执行以下SQL语句：

SELECT name, mfgr, retailprice, supplycost

FROM part JOIN partsupp ON part.partkey = partsupp.partkey

WHERE retailprice > supplycost \* 10000;

部分结果截屏如下：



（8）三表连接查询

查询顾客姓名为“阿波罗”订购的订单编号、总价及其订购的零件编号、数量和明细价格。

执行以下SQL语句：

SELECT orders.orderkey, orders.totalprice, lineitem.partkey, lineitem.linenumber, lineitem.extendedprice

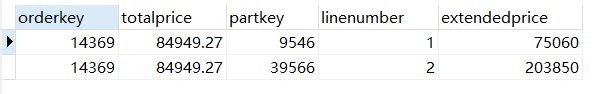
FROM orders

JOIN lineitem ON orders.orderkey=lineitem.orderkey

JOIN part ON part.partkey=lineitem.partkey

WHERE custkey = (SELECT custkey FROM customer WHERE name='阿波罗')

部分结果截屏如下：



* + - 1. **与实验结果相关的文件**

无

* + - 1. **实验总结**

这次的 SQL 实验让我更深入地理解了数据库查询和编程规范。通过设计各种查询语句，包括单表查询、分组统计查询和连接查询，我不仅掌握了 SQL 语言的基本用法，还学会了如何合理地组织和优化查询，以提高数据库的效率。

实验中的重点是分组统计查询和连接查询，这些技能在实际数据库应用中非常有用。我学会了区分元组过滤条件和分组过滤条件，这对于正确编写复杂的查询语句至关重要。同时，确定连接属性和设计连接条件也是实验的难点，但通过不断的练习，我逐渐掌握了这些技巧。

最重要的是，我学会了遵循 SQL 程序设计规范，包括关键字的大小写、标识符的命名规范等。这将有助于我的 SQL 编程更加规范简洁。

总的来说，这次实验让我在 SQL 编程方面取得了显著的进步，我相信这些知识和技能将在未来的数据库应用中发挥重要作用