**《数据库系统原理》**

**课程实验报告**



**学院：计算机（国家示范性软件）学院**

**班级：** 2021211304

**姓名：** **张梓良 杨晨 苗雨**

**学号：**2021212484 2021212171 2021212492

目录

[1实验目的 3](#_Toc154682999)

[2实验平台及环境 3](#_Toc154683000)

[3实验内容及要求 3](#_Toc154683001)

[3.1实验内容 3](#_Toc154683002)

[3.2实验要求 3](#_Toc154683003)

[4实验步骤 4](#_Toc154683004)

[4.1单表查询 4](#_Toc154683005)

[4.2 String操作 7](#_Toc154683006)

[4.3集合操作 9](#_Toc154683007)

[4.4多表查询 11](#_Toc154683008)

[4.5聚集函数 14](#_Toc154683009)

[4.6嵌套查询 16](#_Toc154683010)

[4.7 with 临时视图查询 22](#_Toc154683011)

[4.8 键/函数依赖分析 23](#_Toc154683012)

[4.9 关系表的插入/删除/更新 25](#_Toc154683013)

[5实验总结 28](#_Toc154683014)

# 1实验目的

对前面实验建立的电商数据库关系表进行各种类型的查询操作和修改操作，加深对SQL语言中DML的了解，掌握相关查询语句和数据修改语句的使用方法。

# 2实验平台及环境

1. 实验平台：
   1. GaussDB(for openGauss)云数据库
   2. ip：116.205.151.208
   3. 数据库：finance
2. 实验环境：Dbeaver 23.2.3
3. 编程语言：SQL

# 3实验内容及要求

## 3.1实验内容

1. 单表简单查询，包括复合选择条件、结果排序、结果去重、结果重命名查询；
2. 多表查询，包括等值连接、自然连接、元组变量查询；
3. 统计查询，包括带有分组、聚集函数的查询；
4. 嵌套查询，包括带有in/some/all、 exists、unique 的嵌套查询，from中子查询；
5. with 临时视图查询；
6. 键/函数依赖分析；
7. 表的插入、删除、更新；

## 3.2实验要求

1. 用 postgreSQL 语句完成以上操作。
2. 从上述29个/组查询中，选择一部分查询，查询语句总数不少于16个。

# 4实验步骤

## 4.1单表查询

**查询1：**从订单表ORDERS表中，找出由收银员Clerk#000000951处理的满足下列条件的所有订单o\_orderkey：

（1）订单总价位于[？,？]，并且

（2）下单日期在？至？之间，并且

（3）订单状态O\_ORDERSTATUS不为空

列出这些订单的订单key（O\_ORDERKEY）、客户key、订单状态、订单总价、下单日期、订单优先级和发货优先级；

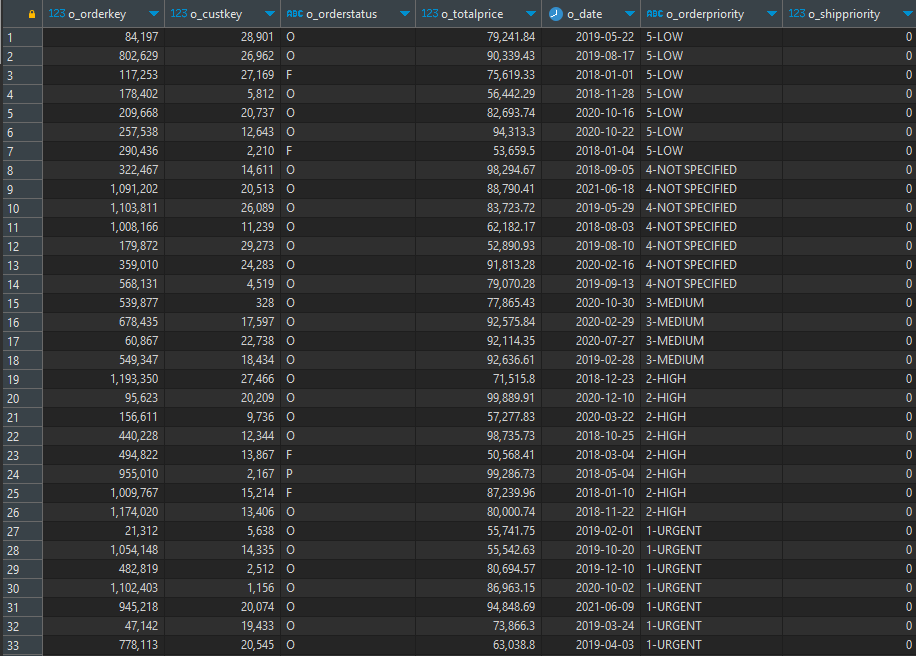
要求：对查询结果，按照订单优先级从高到低、发货优先级从高到低排序，并且将O\_ORDERDATE重新命名为O\_DATE。

说明：？代表由学生自己选择输入条件

**SQL语句**

1. select O\_ORDERKEY, O\_CUSTKEY, O\_ORDERSTATUS, O\_TOTALPRICE, O\_ORDERDATE AS O\_DATE, O\_ORDERPRIORITY, O\_SHIPPRIORITY
2. from ORDERS
3. where O\_CLERK='Clerk#000000951'
4. and O\_TOTALPRICE between 50000 and 100000
5. and O\_ORDERDATE between '2018-01-01'::date and '2021-12-31'::date
6. and O\_ORDERSTATUS is not null
7. order by O\_ORDERPRIORITY desc, O\_SHIPPRIORITY desc;

**查询结果**

****

**查询2：**从订单明细表LINEITEM表中，找出满足下列条件的所有订单L\_ORDERKEY：

（1）数量位于[？,？]，

（2）退货标志为‘N’的订单中，价格不小于？

列出这些订单的key和零件供应商key、价格；要求：对查询结果，按照价格从高到低排序，并且对查询结果使用distinct 去重。

比较对查询结果去重和不去重，在查询时间和查询结果上的差异。

**SQL语句**

**·去重**

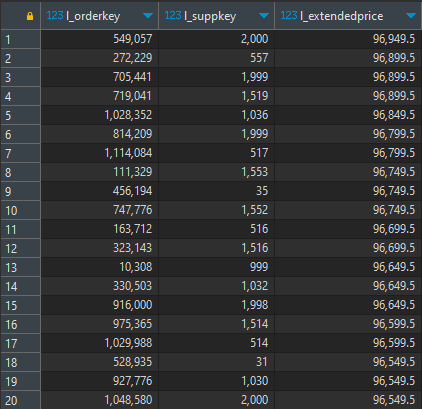
1. select distinct L\_ORDERKEY, L\_SUPPKEY, L\_EXTENDEDPRICE
2. from LINEITEM
3. where L\_QUANTITY between 50 and 100
4. and L\_RETURNFLAG='N'
5. and L\_EXTENDEDPRICE>=50000
6. order by L\_EXTENDEDPRICE desc;

**·不去重**

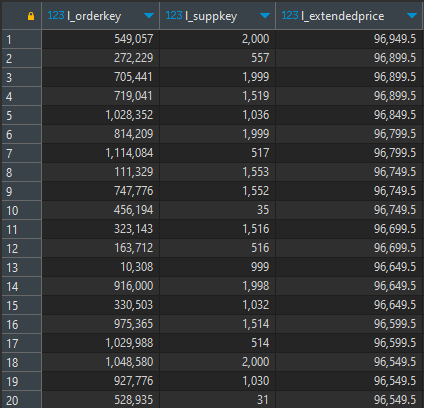
1. select L\_ORDERKEY, L\_SUPPKEY, L\_EXTENDEDPRICE
2. from LINEITEM
3. where L\_QUANTITY between 50 and 100
4. and L\_RETURNFLAG='N'
5. and L\_EXTENDEDPRICE>=50000
6. order by L\_EXTENDEDPRICE desc;

**查询结果**

**·去重**

****

**·不去重**

****

不对查询结果进行去重可能会导致返回重复的条目。对查询结果进行去重可以确保结果集中的每个条目都是唯一的，消除了冗余信息，使结果更加准确和清晰。

**时间对比**

**·去重**

****

**·不去重**

****

如果不对查询结果进行去重，那么系统可能返回重复的结果。这意味着查询时间可能会更短，因为系统只需返回所有匹配的结果，无需额外的去重处理。如果对查询结果进行去重，系统需要花费额外的计算资源来识别和删除重复项。这可能会导致查询时间增加，特别是当查询结果很大时，去重操作可能会耗费较多的时间。

## 4.2 String操作

**查询3：**从客户表CUSTOMER表中，找出满足下列条件的客户：

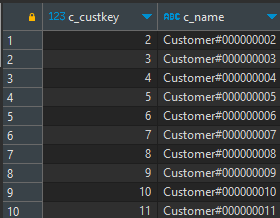
（1）客户电话开头部分包含‘10’，或者客户市场领域中包含“BUILDING”，并且

（2）客户电话结尾不为‘8’。

**SQL语句**

1. select C\_CUSTKEY, C\_NAME
2. from CUSTOMER
3. where (C\_PHONE like '10%' or C\_MKTSEGMENT like '%BUILDING%')
4. and C\_PHONE not like '%8';

**查询结果**

****

**查询4：**从客户表CUSTOMER表中，找出满足下列条件的客户姓名：

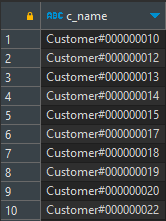
（1）客户key由2个字符组成，并且

（2）客户地址至少包括18个字符，即地址字符串的长度不小于18。

**SQL语句**

1. select C\_NAME
2. from CUSTOMER
3. where C\_CUSTKEY like '\_\_'
4. and length(C\_ADDRESS)>=18;

**查询结果**

****

## 4.3集合操作

**查询5：**使用集合并操作union、union all，从订单明细表LINEITEM查询满足下列条件的订单key

(1)订单发货日期早于‘2016-01-01’，或者

(2)订单数量大于100

对比union all、union 操作在查询结果、执行时间上的差异。

**SQL语句**

**·union all**

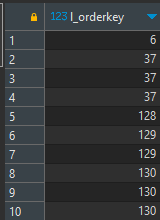
1. select L\_ORDERKEY
2. from LINEITEM
3. where L\_SHIPDATE< '2016-01-01'::date
4. union all
5. select L\_ORDERKEY
6. from LINEITEM
7. where L\_QUANTITY>100;

**·union**

1. select L\_ORDERKEY
2. from LINEITEM
3. where L\_SHIPDATE< '2016-01-01'::date
4. union
5. select L\_ORDERKEY
6. from LINEITEM
7. where L\_QUANTITY>100;

**查询结果**

**·union all**

****

**·union**

****

在查询结果方面，UNION 会去除重复的行，因此查询结果中不会包含重复的行。而 UNION ALL 则会保留所有的行，查询结果中可能会包含重复的行。

**时间对比**

**·union all**

****

**·union**

****

在执行时间方面，由于 UNION 需要去除重复的行，因此它需要进行额外的操作，比 UNION ALL 更耗时。

**查询6：**结合教材3.4.1节元组变量样例，使用集合操作except、except all，从供应商表SUPPLIER中，查询 账户余额最大的供应商。 对比使用except、except all、聚集函数max，对比完成此查询在执行时间、查询结果上的异同。

**SQL语句**

1. *-- except*
2. select S\_SUPPKEY, S\_NAME
3. from SUPPLIER
4. except(
5. select t1. S\_SUPPKEY, t1. S\_NAME
6. from SUPPLIER t1, SUPPLIER t2
7. where t1. S\_ACCTBAL<t2. S\_ACCTBAL
8. );
9. *-- except all*
10. select S\_SUPPKEY, S\_NAME
11. from SUPPLIER
12. except all(
13. select t1. S\_SUPPKEY, t1. S\_NAME
14. from SUPPLIER t1, SUPPLIER t2
15. where t1. S\_ACCTBAL<t2. S\_ACCTBAL
16. );
17. *-- max*
18. select S\_SUPPKEY, S\_NAME
19. from SUPPLIER
20. where S\_ACCTBAL=(
21. select max(S\_ACCTBAL)
22. from SUPPLIER
23. );

**查询结果（均相同）**

****

EXCEPT和EXCEPT ALL这两个操作符都用于比较两个查询的结果集，并返回第一个查询结果中存在但在第二个查询结果中不存在的行。EXCEPT会进行去重操作，因此返回的结果集中不包含重复行；而EXCEPT ALL则返回所有不同的行，可能包括重复行。

MAX聚合函数只返回一列中的最大值，而不涉及行级别的比较或去重操作。因此，它返回的结果只是一个单一的最大值。

**时间对比**

**·except**

****

**·except all**

****

**·max**

****

EXCEPT操作符用于从查询结果中删除第一个查询的结果集中包含的任何行，而后续查询的结果将返回。这涉及到对结果集进行排序和去重操作，因此可能会导致较长的执行时间。

EXCEPT ALL操作符与EXCEPT类似，但不进行去重操作。它直接返回两个查询结果的差异，而无需排序和去重，因此相对较快。

MAX是一个聚合函数，用于返回一列中的最大值。它只需要扫描一次数据，找到最大值并返回，因此通常比EXCEPT操作更快。

## 4.4多表查询

**查询7：** 选取两张数据量比较小的表T1和T2，如地区表REGION、国家表NATION、供应商表SUPPLIER，执行如下无连接条件的笛卡尔积操作，观察数据库系统的反应和查询结果：

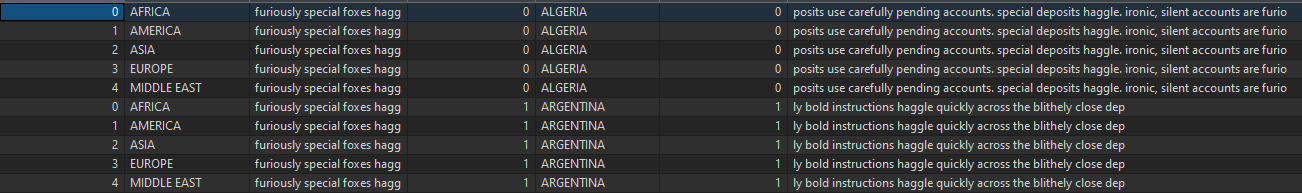
SELECT\*

FROM T1，T2

**SQL语句**

1. select\*
2. from REGION, NATION;

**查询结果**

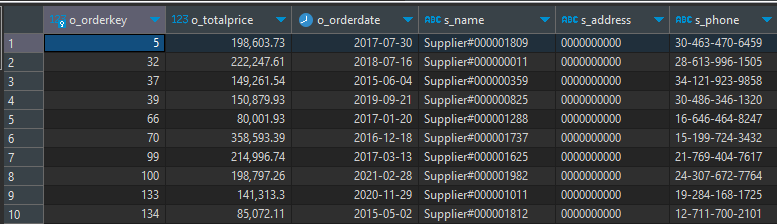
****

**查询8：** 使用多表连接操作（3.3.3 join/natural join，4.1.1 join），从订单表ORDERS、供应商表SUPPLIER、订单明细表LINEITEM中，查询实际到达日期小于预计到达日期的订单，列出这些订单的订单key、订单总价、下单日期以及该供应商的姓名、地址和手机号。

**SQL语句**

1. select O\_ORDERKEY, O\_TOTALPRICE, O\_ORDERDATE, S\_NAME, S\_ADDRESS, S\_PHONE
2. from (LINEITEM join ORDERS on L\_ORDERKEY=O\_ORDERKEY)
3. join SUPPLIER on S\_SUPPKEY= L\_SUPPKEY
4. where L\_RECEIPTDATE < L\_COMMITDATE;

**查询结果**

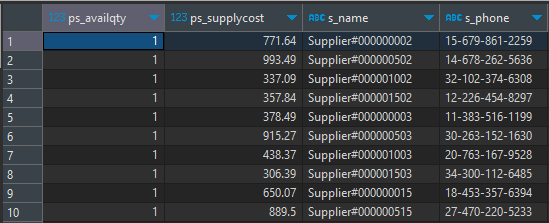


**查询9：** 使用多表连接操作，从供应商表SUPPLIER、零部件表PART、零部件供应表PARTSUPP中，查询供应零件品牌为‘Brand#13’的供应商信息，列出零件供应数量与成本，以及供应商的姓名与手机号。

**SQL语句**

1. select PS\_AVAILQTY, PS\_SUPPLYCOST, S\_NAME, S\_PHONE
2. from PARTSUPP t1 join SUPPLIER on t1.PS\_SUPPKEY = SUPPLIER.S\_SUPPKEY, PART
3. where t1.PS\_PARTKEY= P\_PARTKEY and PART.P\_BRAND='Brand#13';

**查询结果**

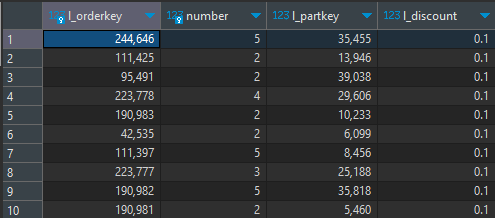
****

**查询10：** 利用订单明细表LINEITEM，使用教材3.4.1节元组变量as/rename方式，查询所有比流水号为“1”，订单号为“1”的折扣高的订单key和流水号，列出这些订单的零件、折扣，结果按照折扣的降序排列。

**SQL语句**

1. select L\_ORDERKEY, L\_LINENUMBER as number, L\_PARTKEY, L\_DISCOUNT
2. from LINEITEM
3. where L\_DISCOUNT>(
4. select L\_DISCOUNT
5. from LINEITEM
6. where L\_LINENUMBER='1'
7. and L\_ORDERKEY='1'
8. )
9. order by L\_DISCOUNT desc;

**查询结果**

****

## 4.5聚集函数

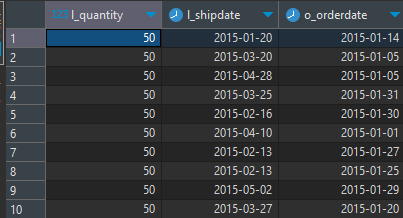
**查询11：**从订单明细表LINEITEM、订单表ORDERS、客户表CUSTOMER、国家表NATION，查询客户来自 ALGERIA，下单日期为'2015-01-01'到'2015-02-02'的订单下列信息： （1）满足条件订单的最大数量、最小数量和平均数量。 （2）具有最大数量且满足上述条件的订单，列出该订单的发货日期、下单日期。

**SQL语句**

1. select max(L\_QUANTITY),min(L\_QUANTITY),avg(L\_QUANTITY)
2. from LINEITEM, CUSTOMER, NATION, ORDERS
3. where LINEITEM. L\_ORDERKEY=ORDERS. O\_ORDERKEY
4. and ORDERS. O\_CUSTKEY= CUSTOMER. C\_CUSTKEY
5. and CUSTOMER. C\_NATIONKEY= NATION. N\_NATIONKEY
6. and NATION. N\_NAME='ALGERIA'
7. and ORDERS. O\_ORDERDATE between '2015-01-01'::date and '2015-02-02'::date;
8. select L\_QUANTITY, L\_SHIPDATE, O\_ORDERDATE
9. from LINEITEM, CUSTOMER, NATION, ORDERS
10. where LINEITEM. L\_ORDERKEY=ORDERS. O\_ORDERKEY
11. and ORDERS. O\_CUSTKEY= CUSTOMER. C\_CUSTKEY
12. and CUSTOMER. C\_NATIONKEY= NATION. N\_NATIONKEY
13. and NATION. N\_NAME='ALGERIA'
14. and ORDERS. O\_ORDERDATE between '2015-01-01'::date and '2015-02-02'::date
15. and L\_QUANTITY = (select max(L\_QUANTITY)
16. from LINEITEM, CUSTOMER, NATION, ORDERS
17. where LINEITEM. L\_ORDERKEY=ORDERS. O\_ORDERKEY
18. and ORDERS. O\_CUSTKEY= CUSTOMER. C\_CUSTKEY
19. and CUSTOMER. C\_NATIONKEY= NATION. N\_NATIONKEY
20. and NATION. N\_NAME='ALGERIA'
21. and ORDERS. O\_ORDERDATE between '2015-01-01'::date and '2015-02-02'::date);

**查询结果**

****

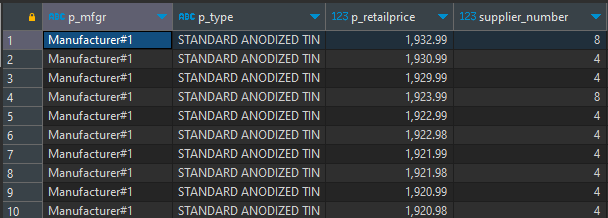
****

**查询12：**根据零部件表PART和零部件供应表PARTSUPP和供应商表SUPPLIER，查询有多少零件厂商提供了品牌为Brand#13的零件，给出这些零件的类型、零售价和供应商数量，并将查询结果按照零售价降序排列。

**SQL语句**

1. select P\_MFGR, P\_TYPE, P\_RETAILPRICE, count(s\_suppkey) as supplier\_number
2. from PART, PARTSUPP, SUPPLIER
3. where PARTSUPP. PS\_PARTKEY=PART. P\_PARTKEY
4. and PARTSUPP.PS\_SUPPKEY=S\_SUPPKEY
5. and PART.P\_BRAND='Brand#13'
6. group by P\_MFGR, P\_TYPE, P\_RETAILPRICE
7. order by P\_RETAILPRICE desc;

**查询结果**

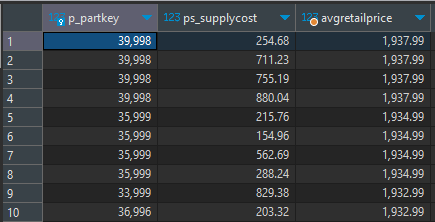
****

**查询13：**从零部件表PART和零部件供应表PARTSUPP中，查询所有零件大小在[7,14]之间的零件的平均零售价，给出零件key，供应成本，平均零售价，结果按照零售价降序排列。

**SQL语句**

1. select P\_PARTKEY, PS\_SUPPLYCOST, avg(P\_RETAILPRICE) as avgretailprice
2. from PART, PARTSUPP
3. where PART. P\_PARTKEY=PARTSUPP. PS\_PARTKEY
4. and PART. P\_SIZE between 7 and 14
5. group by P\_PARTKEY, PS\_SUPPLYCOST
6. order by avgretailprice desc;

**查询结果**



## 4.6嵌套查询

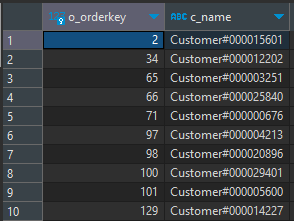
**查询14：**从订单明细表LINEITEM、订单表ORDERS、客户表CUSTOMER中，使用set membership运算符in，查询明细折扣小于0.01的订单，列出这些订单的key和采购订单的客户姓名。 对比使用多表连接、非嵌套的查询在执行时间、查询结果上的异同。

**SQL语句**

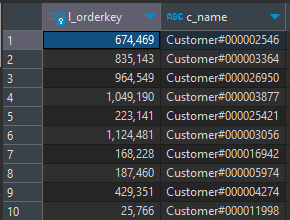
1. *-- 嵌套查询*
2. select O\_ORDERKEY, C\_NAME
3. from ORDERS, CUSTOMER
4. where O\_CUSTKEY= C\_CUSTKEY
5. and O\_ORDERKEY in(
6. select L\_ORDERKEY
7. from LINEITEM
8. where L\_DISCOUNT<0.01);
9. *-- 多表查询*
10. select distinct L\_ORDERKEY, C\_NAME
11. from LINEITEM, ORDERS, CUSTOMER
12. where L\_ORDERKEY= O\_ORDERKEY
13. and O\_CUSTKEY=C\_CUSTKEY
14. and L\_DISCOUNT<0.01;

**查询结果**

**·嵌套查询**

****

**·多表查询**

****

**时间对比**

**·嵌套查询**

****

**·多表查询**

****

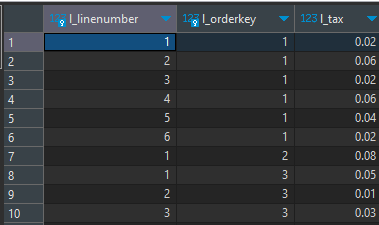
多表连接的查询需要对多个表进行关联操作，因此会花费更多的时间。

**查询15-1：**从订单明细表LINEITEM，使用Set Comparison 运算符some，查询满足下列条件的订单：该订单的数量大于发货日期在[?,?]之间的部分（至少一个）订单的数量，列出这些订单的流水号、key和税。

**SQL语句**

1. select L\_LINENUMBER,L\_ORDERKEY,L\_TAX
2. from LINEITEM
3. where L\_QUANTITY>some(
4. select L\_QUANTITY
5. from LINEITEM
6. where L\_SHIPDATE>= '2019-01-01'::date
7. and L\_SHIPDATE<= '2020-02-02'::date
8. );

**查询结果**

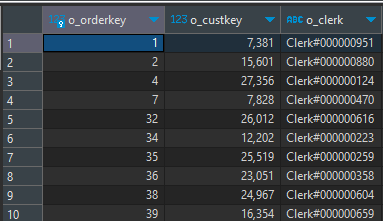


**查询15-2：**从订单表ORDERS，使用Set Comparison 运算符some，查询满足下列条件的订单：订单状态为‘O’，订单总价大于部分在2020年之后下单的订单。列出这些订单的key、客户key、收银员。

**SQL语句**

1. select O\_ORDERKEY, O\_CUSTKEY, O\_CLERK
2. from ORDERS
3. where O\_TOTALPRICE>some(
4. select O\_TOTALPRICE
5. from ORDERS
6. where O\_ORDERDATE>= '2020-01-01'::date
7. ) and O\_ORDERSTATUS='O';

**查询结果**



**查询16-1：**从订单明细表LINEITEM中，使用Set Comparison 运算符>=all，查询满足下列条件的供应商： 该供应商在2019年出货量大于等于同时段其他供应商的出货量，即2019年该供应商的出货量最高。

**SQL语句**

1. select L\_SUPPKEY
2. from LINEITEM
3. where L\_SHIPDATE>= '2019-01-01'::date
4. and L\_SHIPDATE<= '2019-12-30'::date
5. group by L\_SUPPKEY
6. having sum(L\_QUANTITY)>=all(
7. select sum(L\_QUANTITY)
8. from LINEITEM
9. where L\_SHIPDATE>='2019-01-01'::date
10. and L\_SHIPDATE<='2019-12-30'::date
11. group by L\_SUPPKEY
12. );

**查询结果**



**查询16-2：**供应商表SUPPLIER，使用Set Comparison 运算符all，查询账户余额大于等于其他供应商的供应商。列出该供应商的姓名、key、手机号。

**SQL语句**

1. select S\_SUPPKEY, S\_NAME, S\_PHONE
2. from SUPPLIER
3. where S\_ACCTBAL>=all(
4. select S\_ACCTBAL
5. from SUPPLIER
6. );

**查询结果**

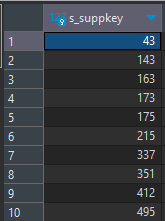


**查询17-1：**从供应商表SUPPLIER、国家表NATION，使用Test for Empty Relations运算符“exists”，查询国家为日本，账户余额大于5000的供应商。

**SQL语句**

1. select SUPPLIER. S\_SUPPKEY
2. from SUPPLIER
3. where exists(
4. select\*
5. from NATION
6. where NATION. N\_NATIONKEY= SUPPLIER. S\_NATIONKEY
7. and NATION .N\_NAME='JAPAN'
8. and SUPPLIER.S\_ACCTBAL>5000
9. );

**查询结果**



**查询17-2：**从客户表CUSTOMER、国家表NATION、订单表ORDERS、订单明细表LINEITEM、供应商表SUPPLIER 中，使用Test for Empty Relations 运算符“not exists except”，查询满足下列条件的供应商：该供应商能供应所有的零件。（本题对指导书所给题目进行了修改，因为所有供应商都不能供应所有零件）

**SQL语句**

1. select S\_SUPPKEY
2. from SUPPLIER
3. where not exists(
4. select P\_PARTKEY
5. from PART
6. except(
7. select PS\_PARTKEY
8. from PARTSUPP
9. where PS\_SUPPKEY= S\_SUPPKEY
10. ));

**查询结果**



**查询18：**从国家表NATION、客户表CUSTOMER中，使用“count”，查询满足下列条件的国家：至少有3个客户来自这个国家，并列出该国家的国家key和国家名。

**SQL语句**

1. select N\_NATIONKEY, N\_NAME
2. from NATION, CUSTOMER
3. where NATION. N\_NATIONKEY=CUSTOMER. C\_NATIONKEY
4. group by N\_NATIONKEY
5. having count(C\_CUSTKEY)>=3;

**查询结果**

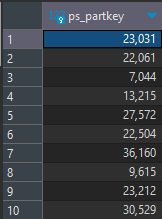


**查询19：**从零部件表PART和零部件供应表PARTSUPP中，使用Subqueries in the From Clause方法，查询满足下列条件的零件：零件由2个以上的供应商供应，且零件大小在20以上。

**SQL语句**

1. select PS\_PARTKEY
2. from(
3. select PS\_PARTKEY, P\_SIZE
4. from PART, PARTSUPP
5. where PART. P\_PARTKEY= PARTSUPP. PS\_PARTKEY
6. group by PS\_PARTKEY, P\_SIZE
7. having count(PS\_SUPPKEY)>2
8. )
9. where P\_SIZE>20;

**查询结果**



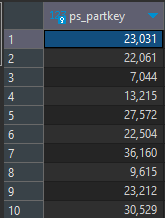
## 4.7 with 临时视图查询

**查询20：**用with临时视图方式，实现查询19中查询要求。

**SQL语句**

1. with temp as(
2. select PS\_PARTKEY, P\_SIZE
3. from PART, PARTSUPP
4. where PART. P\_PARTKEY= PARTSUPP. PS\_PARTKEY
5. group by PS\_PARTKEY, P\_SIZE
6. having count(PS\_SUPPKEY)>2
7. )
8. select PS\_PARTKEY
9. from temp
10. where P\_SIZE>20;

**查询结果**

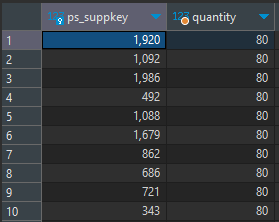


**查询21：**从零部件供应表PARTSUPP中，用with临时视图方式，查询零件供应数量最多的供应商key和其供应的数量。

**SQL语句**

1. with supnum as (
2. select PS\_SUPPKEY, sum(PS\_AVAILQTY) as quantity
3. from PARTSUPP
4. group by PS\_SUPPKEY
5. )
6. select PS\_SUPPKEY, quantity
7. from supnum
8. where quantity=(select max(quantity)
9. from supnum)

**查询结果**



## 4.8 键/函数依赖分析

**查询22：**在订单明细表LINEITEM中，检查订单key、零件key、供应商key、流水号是否组成超键。

**SQL语句**

1. select L\_ORDERKEY, L\_PARTKEY, L\_SUPPKEY, L\_LINENUMBER
2. from LINEITEM
3. group by L\_ORDERKEY, L\_PARTKEY, L\_SUPPKEY, L\_LINENUMBER
4. having count(\*)>1;

**查询结果**



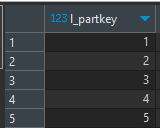
查询结果为空，说明订单key、零件key、供应商key、流水号组成超键

**查询23：**在订单明细表LINEITEM中，利用SQL语句检查函数依赖零件key→价格是否成立；如果不成立，利用SQL语句找出导致函数依赖不成立的元组。

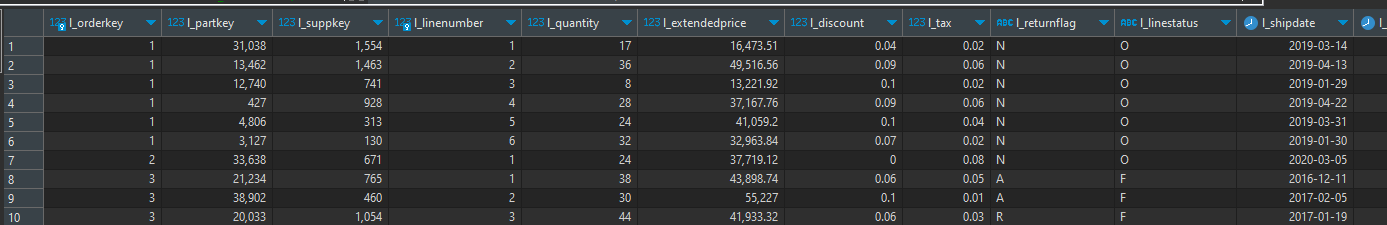
**SQL语句**

1. *-- 判断函数依赖是否成立*
2. select L\_PARTKEY
3. from LINEITEM
4. group by L\_PARTKEY
5. having count(distinct L\_EXTENDEDPRICE)>1;
6. *-- 找出导致函数依赖不成立的元组*
7. select \*
8. from LINEITEM a
9. where a. L\_PARTKEY in(
10. select b. L\_PARTKEY
11. from LINEITEM b
12. group by b. L\_PARTKEY
13. having count(distinct L\_EXTENDEDPRICE)>1
14. );

**查询结果**



查询结果不为空，函数依赖不成立，下面是导致函数依赖不成立的元组：



## 4.9 关系表的插入/删除/更新

**查询24：**向订单表ORDERS中插入一条订单数据。

**SQL语句**

1. insert into ORDERS
2. values('1200001','20045','F',61365.24,'2017-03-19'::date,
3. '2-HIGH','Clerk#000000098',0,'furiously special f');
4. select \*
5. from ORDERS
6. where o\_orderkey = '1200001'

**查询结果**



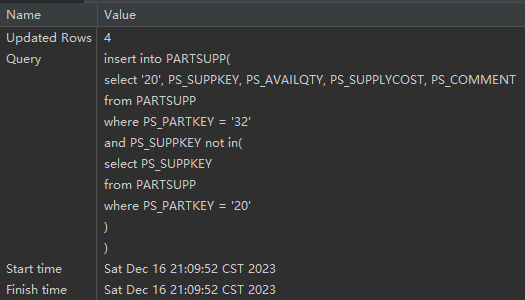
通过查询结果可知插入成功。

**查询25：**将零件32的全部供应商，作为零件20的供应商，加入到零部件供应表PARTSUPP中。

**SQL语句**

1. insert into PARTSUPP(
2. select '20', PS\_SUPPKEY, PS\_AVAILQTY, PS\_SUPPLYCOST, PS\_COMMENT
3. from PARTSUPP
4. where PS\_PARTKEY = '32'
5. and PS\_SUPPKEY not in(
6. select PS\_SUPPKEY
7. from PARTSUPP
8. where PS\_PARTKEY = '20'
9. )
10. );

**查询结果**

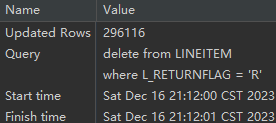


**查询26：**在订单明细表LINEITEM中，删除已退货的订单记录。(returnflag='R')

**SQL语句**

1. delete from LINEITEM
2. where L\_RETURNFLAG = 'R';

**查询结果**

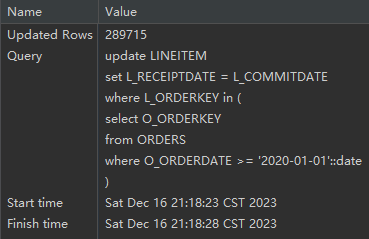
****

**查询27：**用订单明细表LINEITEM中在2019年之后交易中的预计到达日期，替换表中的实际到达日期。

**SQL语句**

1. update LINEITEM
2. set L\_RECEIPTDATE = L\_COMMITDATE
3. where L\_ORDERKEY in (
4. select O\_ORDERKEY
5. from ORDERS
6. where O\_ORDERDATE >= '2020-01-01'::date
7. );

**查询结果**

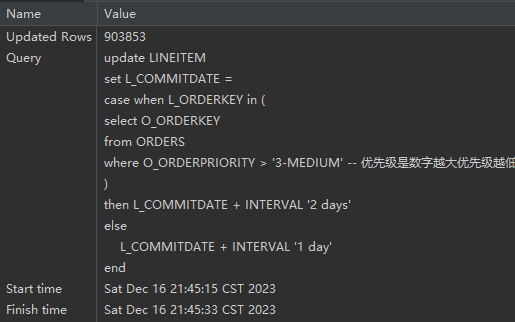
****

**查询28：**针对订单明细表LINEITEM、订单表ORDERS，使用update/case语句做出如下修改：如果订单的订单优先级低于medium，则其在订单明细表中的预计到达日期推后2天, 否则推迟一天。

**SQL语句**

1. update LINEITEM
2. set L\_COMMITDATE =
3. case when L\_ORDERKEY in (
4. select O\_ORDERKEY
5. from ORDERS
6. where O\_ORDERPRIORITY > '3-MEDIUM' *-- 优先级是数字越大优先级越低*
7. )
8. then L\_COMMITDATE + INTERVAL '2 days'
9. else
10. L\_COMMITDATE + INTERVAL '1 day'
11. end;

**查询结果**

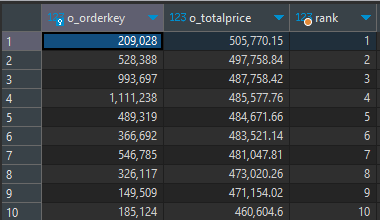
****

**查询29：**在订单表ORDERS中，按照订单总价对订单进行降序排序，并输出订单key和排名。

**SQL语句**

1. select O\_ORDERKEY ,O\_TOTALPRICE,rank() over(order by O\_TOTALPRICE desc) as Rank
2. from ORDERS;

**查询结果**

****

# 5实验总结

在单表简单查询中，我们学习了如何使用选择条件、排序和重命名查询来检索单个表中的数据。我们还学习了如何去重复值，以确保结果集不包含重复的记录。

在多表查询中，我们学习了如何使用等值连接、自然连接和元组变量查询来检索多个相关表中的数据。

在统计查询中，我们学习了如何使用聚合函数和分组操作来汇总数据，并在结果集中显示统计信息。

在嵌套查询中，我们学习了如何使用IN/SOME/ALL、EXISTS和UNIQUE等操作符来创建嵌套查询。我们还学习了如何使用FROM中的子查询来检索数据。

在with临时视图查询中，我们学习了如何使用WITH子句创建临时视图，以在查询中使用它们。

在键/函数依赖分析中，我们学习了如何分析关系模式中的依赖性，以便更好地设计数据库结构。

最后，我们还学习了如何使用SQL语句来插入、删除和更新数据库中的数据。

综上所述，本次实验让我们更好地理解了SQL查询的基础知识，并帮助我们学会了如何使用SQL对数据库进行操作。这些技能在任何数据库领域都是非常重要的，因为它们使我们能够高效地检索、分析和管理数据。