第三章　第三次作业

1. 在 SQL 中，如果在 SELECT 语句中使用了未被聚合的属性，那么这些属性必须如何处理？

   A. 它们可以随意出现，不需要在 GROUP BY 中列出

   B. [✓ 它们必须在 GROUP BY 子句中出现]

   C. 它们可以出现在 HAVING 子句中，而不需要在 GROUP BY 中

   D. 它们只能出现在 WHERE 子句中

2. WHERE 和 HAVING 子句的主要区别是什么？

   A. WHERE 子句只能用于数值列，而 HAVING 子句可以用于所有类型的列

   B. [✓ WHERE 子句在数据分组之前过滤数据，而 HAVING 子句在数据分组之后过滤聚合结果]

   C. HAVING 子句只能在没有 GROUP BY 的情况下使用，而 WHERE 子句可以在有 GROUP BY 的情况下使用

   D. WHERE 子句和 HAVING 子句的功能完全相同

3. 标量子查询的主要特征是什么？

   A. 返回多个行和多个列的结果

   B. [✓ 返回一个单一的值，通常是一个属性的聚合结果]

   C. 只能在 INSERT 语句中使用

   D. 只能用于 WHERE 子句中

4. 在 MySQL 中，为什么在执行插入操作之前先执行 SELECT FROM WHERE 查询非常重要？

   A. 以便查看表的结构和数据类型

   B. [✓ 以确保要插入的数据不会导致主键或唯一性约束冲突]

   C. 以便快速获取插入操作的执行时间

   D. 以便在插入后自动更新相关的其他表

5. 在 SQL 中，以下关于 EXISTS 和 NOT EXISTS 的描述哪项是正确的？

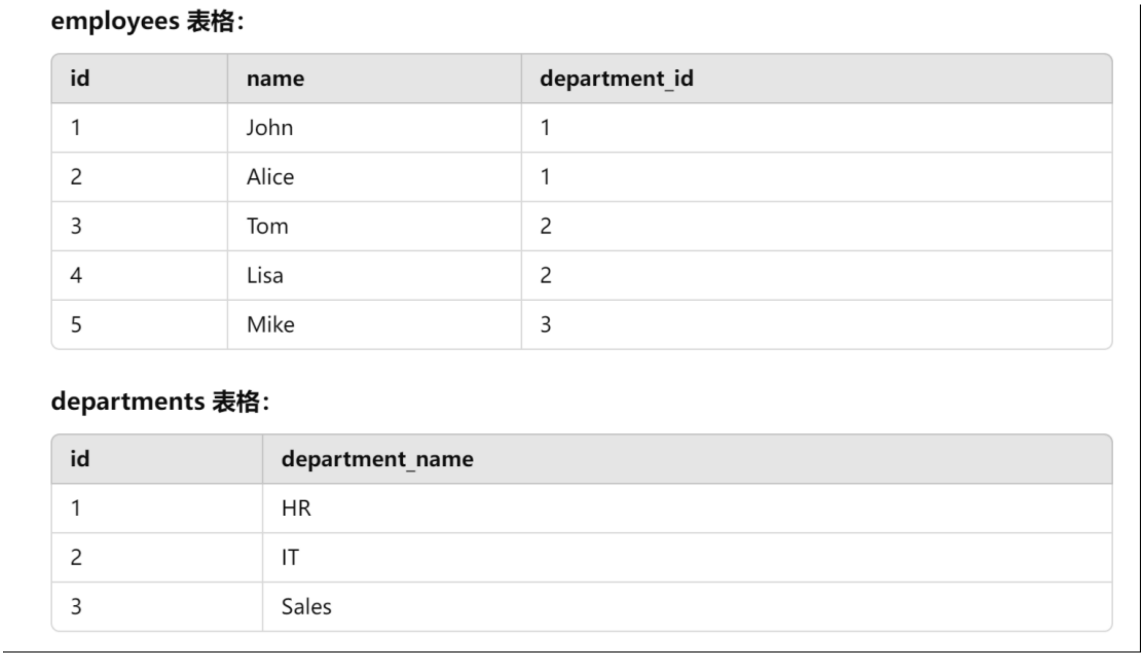
   A. EXISTS 总是返回 TRUE，无论子查询是否返回结果。

   B. [✓ EXISTS r 为 TRUE 当且仅当子查询 r 返回至少一行结果]

   C. NOT EXISTS r 为 TRUE 当且仅当子查询 r 返回至少一行结果。

   D. EXISTS 子查询可以返回多个列，但必须返回唯一的值。

6. 假设有两个表格，”employees” 和”departments”，它们的结构如下。我们想要查询每个部门的员工人数。以下选项中，哪个是合适的嵌套子查询语句？



employees 表格：

id	name	department_id
1	John	1
2	Alice	1
3	Tom	2
4	Lisa	2
5	Mike	3

departments 表格：

id	department_name
1	HR
2	IT
3	Sales

A.

```
SELECT department_name, COUNT(*) FROM employees JOIN departments ON
↪  employees.department_id = departments.id;
```

B.

```
SELECT department_name, COUNT(*) FROM employees WHERE department_id
↪  = departments.id;
```

C.

```
SELECT department_name, (SELECT COUNT(*) FROM employees WHERE
↪  employees.department_id = departments.id) FROM departments;
```

D.

```
SELECT department_name, COUNT(*) FROM departments JOIN employees ON
↪  employees.department_id = departments.id;
```

验证一下

```sql
drop table if exists employees, departments;

create table employees(
    id int auto_increment primary key ,
    name varchar(10),
    department_id varchar(10)
);
create table departments(
    id int auto_increment primary key ,
    department_name varchar(10)
);

insert into employees(name, department_id)
values ('John', 1),
       ('Alice', 1),
       ('Tom', 2),
       ('Lisa', 2),
       ('Mike', 3);

insert into departments(department_name)
values ('HR'),
       ('IT'),
       ('Sales');

SELECT department_name, (SELECT COUNT(*) FROM employees WHERE
↪   employees.department_id = departments.id) FROM departments;
```

	department_name	(SELECT ...)
1	HR	2
2	IT	2
3	Sales	1

7. count(1)、count(*) 与 count(列名) 的区别?

在执行效果上，count(1) 和 count(*) 都会把空值算入结果，而 count(列名) 会忽略空值。

在执行效率上，count(主键列名) > count(1) > count(非主键列名)。

8. Mysql 中 exist 和 in 的区别?

参考 https://blog.csdn.net/jinjiniao1/article/details/92666614 https://cloud.tencent.com/developer/article/1144244 https://cloud.tencent.com/developer/article/1144253

那么可以考虑以下两条语句：

```
select * from A where exists (select * from B where B.id = A.id);
```

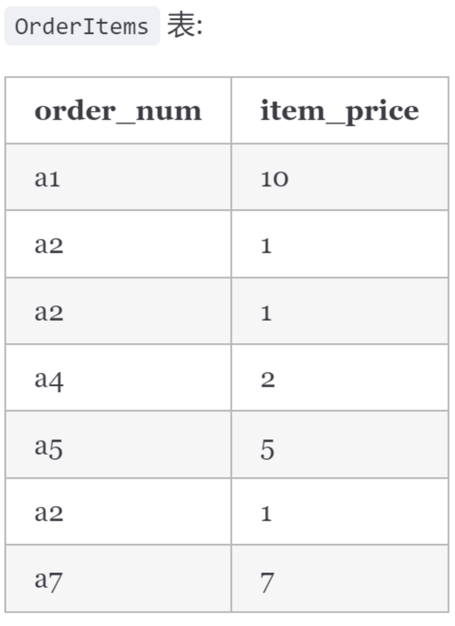
```
select * from A where A.id in (select id from B);
```

以下称 A 为内表，B 为外表，两条语句中括号内的内容为子查询结果。

根据参考文章可知，exists 语句的执行过程是先把内表的数据全部取出来，然后对每一条数据判断是否满足子查询的条件，只用到了子查询的索引；而 in 语句的执行过程是先将子查询结果查询出来，再和内表连接，用到了内表和外表的索引。

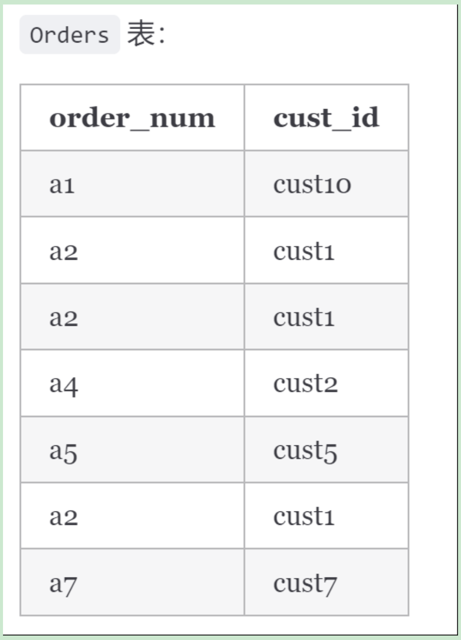
所以当子查询结果的数据较多，外表的数据较少时，exists 有 Block 嵌套循环优化（目前还不理解），查询效率更高；而子查询结果的数据较少，外表的数据较多时，in 由于能用到外表的索引，所以效率更高。

9. OrderItems 表示订单商品表，含有字段订单号：order_num，订单价格：item_price；Orders 表代表订单信息表，含有顾客 id：cust_id 和订单号：order_num。使用子查询，返回购买价格为 10 美元或以上产品的顾客列表，结果无需排序。



OrderItems 表:

order_num	item_price
a1	10
a2	1
a2	1
a4	2
a5	5
a2	1
a7	7



Orders 表:

order_num	cust_id
a1	cust10
a2	cust1
a2	cust1
a4	cust2
a5	cust5
a2	cust1
a7	cust7

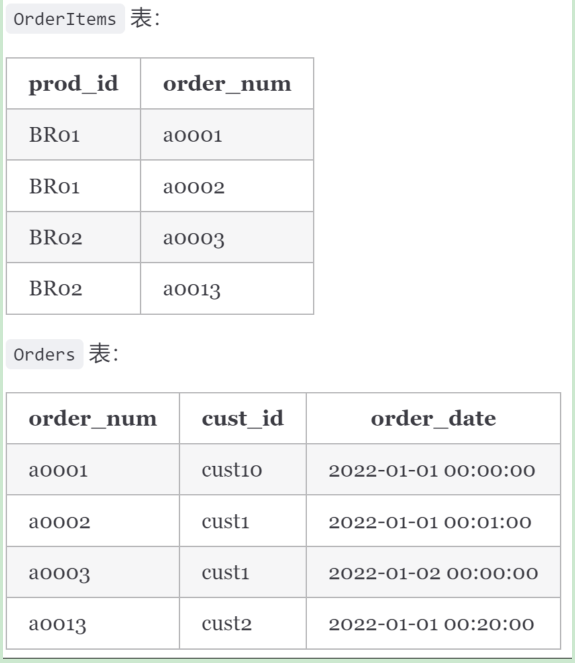
```
select cust_id from Orders where order_num in (
    select order_num from OrderItems where item_price >= 10
);
```

验证一下

```sql
drop table if exists orderitems, orders;

create table if not exists OrderItems
(
    order_num  varchar(10),
    item_price int
);
insert into OrderItems
values ('a1', 10),
       ('a2', 1),
       ('a2', 1),
       ('a2', 1),
       ('a4', 2),
       ('a5', 5),
       ('a2', 1),
       ('a7', 7);

create table if not exists Orders
(
    order_num varchar(10),
    cust_id   varchar(10)
);

insert into Orders
values ('a1', 'cust10'),
       ('a2', 'cust1'),
       ('a2', 'cust1'),
       ('a4', 'cust2'),
       ('a5', 'cust5'),
       ('a2', 'cust4'),
       ('a7', 'cust7');

select cust_id
from Orders
where order_num in (select order_num
                    from OrderItems
                    where item_price >= 10);
```

	cust_id
1	cust10

10. 表 OrderItems 代表订单商品信息表，prod_id 为产品 id；Orders 表代表表订单表有 cust_id 代表顾客 id 和订单日期 order_date。编写 SQL 语句，使用子查询来确定哪些订单（在 OrderItems 中）购买了 prod_id 为”BR01” 的产品，然后从 Orders 表中返回每个产品对应的顾客 ID（cust_id）和订单日期（order_date），按订购日期对结果进行升序排序。



OrderItems 表：

prod_id	order_num
BR01	a0001
BR01	a0002
BR02	a0003
BR02	a0013

Orders 表：

order_num	cust_id	order_date
a0001	cust10	2022-01-01 00:00:00
a0002	cust1	2022-01-01 00:01:00
a0003	cust1	2022-01-02 00:00:00
a0013	cust2	2022-01-01 00:20:00

由于要使用子查询，这里不考虑使用连接，使用 exists 和 in 可以分别写出如下 SQL 语句，其功能一样：

```sql
select cust_id, order_date from Orders where exists(
    select * from OrderItems where prod_id='BR01' and Orders.order_num =
    ↪  OrderItems.order_num
) order by order_date;

select cust_id, order_date from Orders where order_num in (
    select order_num from OrderItems where prod_id='BR01'
) order by order_date;
```

验证一下

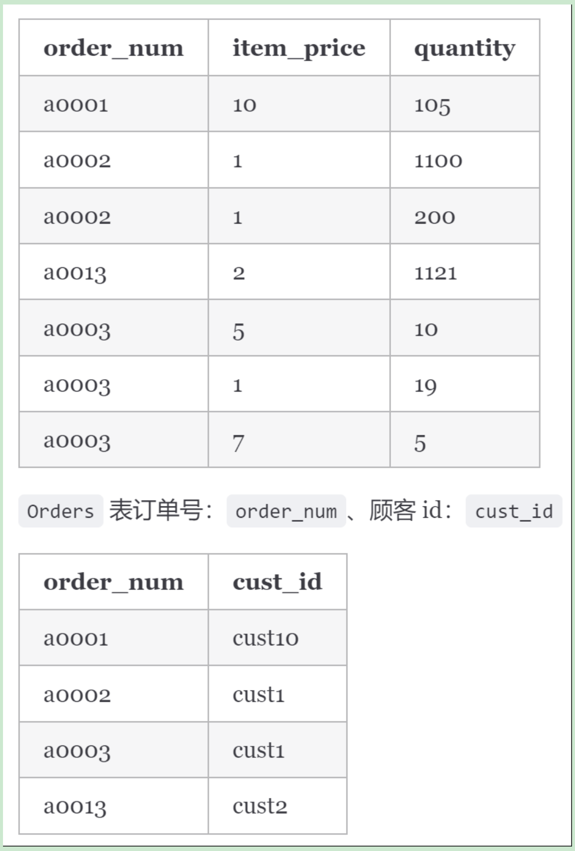
```sql
drop table if exists OrderItems;
create table if not exists OrderItems(
    prod_id varchar(10),
    order_num varchar(10)
);

insert into OrderItems
values ('BR01', 'a0001'),
       ('BR01', 'a0002'),
       ('BR02', 'a0003'),
       ('BR02', 'a0013');

drop table if exists Orders;
create table if not exists Orders(
    order_num varchar(10),
    cust_id varchar(10),
    order_date datetime
);

insert into Orders
values ('a0001', 'cust10', '2022-01-01 00:00:00'),
       ('a0002', 'cust1', '2022-01-01 00:01:00'),
       ('a0003', 'cust1', '2022-01-02 00:00:00'),
       ('a0013', 'cust2', '2022-01-01 00:20:00');
```

以下三种不同语句分别使用 exists、in 和连接，可以得到相同结果。

```sql
select cust_id, order_date from Orders where exists(
    select * from OrderItems where prod_id='BR01' and
     ↪  Orders.order_num = OrderItems.order_num
) order by order_date;

select cust_id, order_date from Orders where order_num in (
    select order_num from OrderItems where prod_id='BR01'
) order by order_date;

select cust_id, order_date from Orders natural join OrderItems where
 ↪  prod_id='BR01' order by order_date;
```

	cust_id	order_date
1	cust10	2022-01-01 00:00:00
2	cust1	2022-01-01 00:01:00

11. 有一个顾客 ID 列表，其中包含他们已订购的总金额。OrderItems 表代表订单信息，OrderItems 表有订单号：order_num 和商品售出价格：item_price、商品数量：quantity。编写 SQL 语句，返回顾客 ID（Orders 表中的 cust_id），并使用子查询返回 total_ordered 每个顾客的所有订单总金额，将结果按金额从大到小排序。



order_num	item_price	quantity
a0001	10	105
a0002	1	1100
a0002	1	200
a0013	2	1121
a0003	5	10
a0003	1	19
a0003	7	5

Orders 表订单号：order_num、顾客 id：cust_id

order_num	cust_id
a0001	cust10
a0002	cust1
a0003	cust1
a0013	cust2

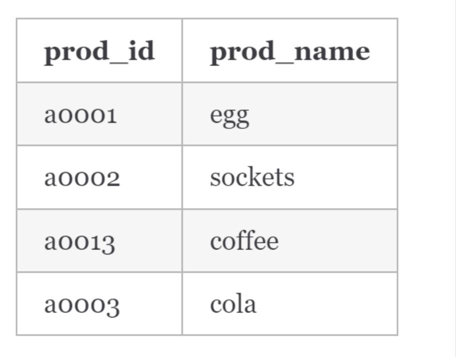
```
select cust_id, sum(total) as total_ordered
from Orders
         join (select order_num, sum(item_price * quantity) as total
               from OrderItems
               group by order_num) as a on Orders.order_num = a.order_num
group by cust_id
order by total_ordered desc;
```

验证一下

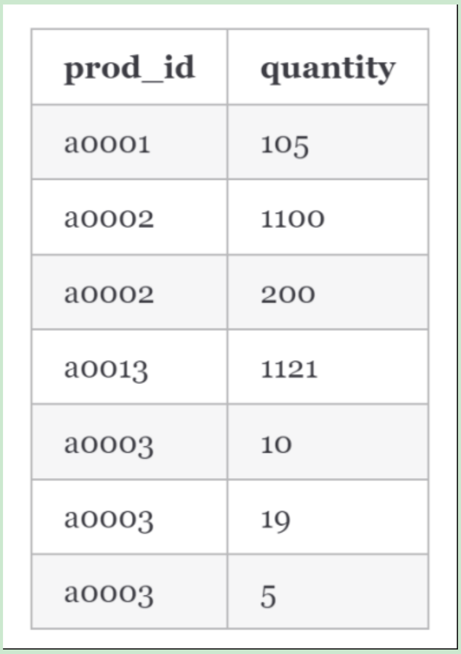
```sql
drop table if exists orderitems, orders;

create table OrderItems(
    order_num varchar(10),
    item_price int,
    quantity int
);
create table Orders(
    order_num varchar(10),
    cust_id varchar(10)
);

insert into OrderItems
values ('a0001', 10, 105),
       ('a0002', 1,1100),
       ('a0002', 1, 200),
       ('a0013', 2, 1121),
       ('a0003', 5, 10),
       ('a0003', 1, 19),
       ('a0003', 7, 5);

insert into Orders
values ('a0001', 'cust10'),
       ('a0002', 'cust1'),
       ('a0003', 'cust1'),
       ('a0013', 'cust2');

select cust_id, sum(total) as total_ordered
from Orders
         join (select order_num, sum(item_price * quantity) as total
               from OrderItems
               group by order_num) as a on Orders.order_num =
               ↪  a.order_num
group by cust_id
order by total_ordered desc;
```

	cust_id	total_ordered
1	cust2	2242
2	cust1	1404
3	cust10	1050

12. Products 表中检索所有的产品名称：prod_name、产品 id：prod_id；OrderItems 代表订单商品表，订单产品：prod_id、售出数量：quantity。编写 SQL 语句，从 Products 表中检索所有的产品名称（prod_name），以及名为 quant_sold 的计算列，其中包含所售产品的总数（在 OrderItems 表上使用子查询和 SUM(quantity) 检索）。



prod_id	prod_name
a0001	egg
a0002	sockets
a0013	coffee
a0003	cola



prod_id	quantity
a0001	105
a0002	1100
a0002	200
a0013	1121
a0003	10
a0003	19
a0003	5

```
select Products.prod_name, a.quant_sold as quant_sold
from Products
         natural join (select prod_id, sum(quantity) as quant_sold
                       from OrderItems
                       group by prod_id) as a;
```

验证一下

```sql
drop table if exists Products, OrderItems;

create table Products (
    prod_id varchar(10),
    prod_name varchar(10)
);
create table OrderItems (
    prod_id varchar(10),
    quantity int
);

insert into Products
values ('a0001', 'egg'),
       ('a0002', 'sockets'),
       ('a0013', 'coffee'),
       ('a0003', 'cola');

insert into OrderItems
values ('a0001', 105),
       ('a0002', 1100),
       ('a0002', 200),
       ('a0013', 1121),
       ('a0003', 10),
       ('a0003', 19),
       ('a0003', 5);

select Products.prod_name, a.quant_sold as quant_sold
from Products
         natural join (select prod_id, sum(quantity) as quant_sold
                       from OrderItems
                       group by prod_id) as a;
```

	prod_name	quant_sold
1	egg	105
2	sockets	1300
3	coffee	1121
4	cola	34