《数据库系统原理》实验报告（）					
题目：实验四					
学号	2351753	姓名	黄保翔	日期	5/1

实验环境：oceanbase，docker

实验步骤及结果截图：

1、建表（见附录一），表内字段的类型可以自行定义（合理即可），注意建表时不要 忽略各表的主键约束和表间的外键约束。

```
obclient(root@sys)[(none)]> create database MOVIE
    -> ;
Query OK, 1 row affected (0.041 sec)

obclient(root@sys)[(none)]> USE MOVIE
Database changed
obclient(root@sys)[MOVIE]> CREATE TABLE Movie(
    ->     Movie_no varchar(10) primary key,
    ->     Movie_name varchar(50),
    ->     Director varchar(30),
    ->     Rating float,
    ->     End_date DATE
    -> );
Query OK, 0 rows affected (0.202 sec)


obclient(root@sys)[MOVIE]> CREATE  TABLE  Viewer
    -> (
    ->     Viewer_no varchar(10) primary key,
    ->     Viewer_name varchar(30),
    ->     Age INT
    -> );
Query OK, 0 rows affected (0.057 sec)

obclient(root@sys)[MOVIE]> CREATE TABLE Watch(
    ->     S_no varchar(10),
    ->     Viewer_no varchar(10),
    ->     Movie_no varchar(10),
    ->     Watch_date DATE,
    ->     primary key (S_no, Viewer_no, Movie_no),
    ->     Foreign key (Movie_no) references Movie(Movie_no) on delete cascade,
    ->     Foreign key (Viewer_no) references Viewer(Viewer_no) on delete cascade
    -> );
Query OK, 0 rows affected (0.111 sec)
```

2. 插入样例数据（见附录二）。

```
obclient(root@sys)[MOVIE]> insert into Movie (Movie_no, Movie_name, Director, Rating, End_date)
    -> values
    -> ('M001', '星际穿越', '克里斯托弗·诺兰', 9.3, '2024-05-01'),
    -> ('M002', '泰坦尼克号', '詹姆斯·卡梅隆', 9.1, '2024-04-10'),
    -> ('M003', '盗梦空间', '克里斯托弗·诺兰', 8.8, '2024-04-20'),
    -> ('M004', '科幻冒险之旅', '张三', 7.5, '2024-04-18'),
    -> ('M005', '爱情故事', '李四', 7.0, '2024-04-25');
Query OK, 5 rows affected (0.022 sec)
Records: 5  Duplicates: 0  Warnings: 0

obclient(root@sys)[MOVIE]> INSERT INTO Viewer (Viewer_no, Viewer_name, Age)
    -> values
    -> ('V001', '李明', 25),
    -> ('V002', '王红', 30),
    -> ('V003', '张磊', 22),
    -> ('V004', '赵颖', 28),
    -> ('V005', '孙杨', 35);
Query OK, 5 rows affected (0.010 sec)
Records: 5  Duplicates: 0  Warnings: 0


obclient(root@sys)[MOVIE]> INSERT INTO Watch
    -> VALUES
    -> ('1', 'V001', 'M001', '2024-03-15'),
    -> ('2', 'V001', 'M001', '2024-03-20'),
    -> ('2', 'V001', 'M002', '2024-03-20'),
    -> ('3', 'V002', 'M002', '2024-03-25'),
    -> ('1', 'V002', 'M003', '2024-04-01'),
    -> ('2', 'V003', 'M001', '2024-04-05'),
    -> ('2', 'V004', 'M002', '2024-04-12'),
    -> ('1', 'V005', 'M003', '2024-04-14');
Query OK, 8 rows affected (0.013 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

3.查询电影名称中包含“科幻”的电影信息，输出所有信息（包括电影名称、电影编 号、导演、评分、电影停映日期），并按照评分降序排列

```
obclient(root@sys)[MOVIE]> SELECT *
    -> FROM Movie
    -> WHERE Movie_name LIKE '%科幻%'
    -> ORDER BY Rating DESC;
+----------+--------------------+----------+--------+------------+
| Movie_no | Movie_name         | Director | Rating | End_date   |
+----------+--------------------+----------+--------+------------+
| M004     | 科幻冒险之旅 | 张三   |    7.5 | 2024-04-18 |
+----------+--------------------+----------+--------+------------+
1 row in set (0.010 sec)
```

4. 查询观看了电影名为“泰坦尼克号”的观众信息，输出该观众的编号、姓名和年龄， 并按照观众编号升序排列。

```
obclient(root@sys)[MOVIE]> SELECT VIEWER.Viewer_no, VIEWER.Viewer_name, VIEWER.Age
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no join Movie on Watch.Movie_no = Movie.Movie_no
    -> WHERE Movie_name = '泰坦尼克号'
    -> ORDER BY Viewer.Viewer_no;
+-----------+-------------+------+
| Viewer_no | Viewer_name | Age  |
+-----------+-------------+------+
| V001      | 李明      |   25 |
| V002      | 王红      |   30 |
| V004      | 赵颖      |   28 |
+-----------+-------------+------+
3 rows in set (0.069 sec)
```

5. 统计每个观众的观影信息，输出每个观众的编号、观看的电影名称和观看日期。

```
obclient(root@sys)[MOVIE]> SELECT Watch.Viewer_no, Movie.Movie_name, Watch.Watch_date
    -> FROM Watch join Movie on Watch.Movie_no = Movie.Movie_no;
+-----------+-----------------+------------+
| Viewer_no | Movie_name      | Watch_date |
+-----------+-----------------+------------+
| V001      | 星际穿越        | 2024-03-15 |
| V001      | 星际穿越        | 2024-03-20 |
| V003      | 星际穿越        | 2024-04-05 |
| V001      | 泰坦尼克号      | 2024-03-20 |
| V004      | 泰坦尼克号      | 2024-04-12 |
| V002      | 泰坦尼克号      | 2024-03-25 |
| V002      | 盗梦空间        | 2024-04-01 |
| V005      | 盗梦空间        | 2024-04-14 |
+-----------+-----------------+------------+
8 rows in set (0.010 sec)
```

6. 查询所有已停映电影的信息，输出观众编号、姓名、电影名称和观看日期，并按观 看日期降序排列。

P.S.已停映电影指的是“现实日期”大于电影停映日期字段的电影，“现实日期” 以 4 月 15 日为例

```
obclient(root@sys)[MOVIE]> SELECT Viewer.Viewer_no, Viewer.Viewer_name, Movie.Movie_name, Watch.Watch_date
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no join Movie on Watch.Movie_no = Movie.Movie_no
    -> WHERE Movie.End_date < '2024-04-15'
    -> ORDER BY Watch.Watch_date DESC;
+-----------+-------------+-----------------+------------+
| Viewer_no | Viewer_name | Movie_name      | Watch_date |
+-----------+-------------+-----------------+------------+
| V004      | 赵颖        | 泰坦尼克号 | 2024-04-12 |
| V002      | 王红        | 泰坦尼克号 | 2024-03-25 |
| V001      | 李明        | 泰坦尼克号 | 2024-03-20 |
+-----------+-------------+-----------------+------------+
3 rows in set (0.011 sec)
```

7.查询观看了“星际穿越”但没有观看“盗梦空间”的观众信息，输出这些观众的编 号，并按照编号升序排列。

```
obclient(root@sys)[MOVIE]> SELECT Viewer.Viewer_no
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no join Movie on Watch.Movie_no = Movie.Movie_no
    -> WHERE Movie_name = '星际穿越'
    -> EXCEPT
    -> (SELECT Viewer.Viewer_no
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no join Movie on Watch.Movie_no = Movie.Movie_no
    -> WHERE Movie_name = '盗梦空间')
    -> ORDER BY Viewer_no;
+-----------+
| Viewer_no |
+-----------+
| V001      |
| V003      |
+-----------+
2 rows in set (0.034 sec)
```

8. 创建一个过程，使之能够实现如下功能：

修改观影表，增加字段”重复观看状态”（字段名为“Repeat_state”），字段含 义为表示某观众是否多次观看某电影；

并根据表中已有数据为该字段赋值（所赋的值与表定义时的数据类型保持一致即可， 比如可以定义多次观看某电影的“重复观看状态”为 True，只看过一次某电影的 “重复观看状态”为 False），要求使用 if 语句进行条件判断。 P.S.创建存储过程的语法可参考如下代码片段，其中存储过程的名称、是否带参数、参数的名称类型自行决定，合理即可；

```
obclient(root@sys)[MOVIE]> DROP PROCEDURE IF EXISTS update_state;
    ->
    -> -- 创建新的存储过程 update_state
    -> CREATE procedure update_state()
    -> begin
    ->     -- 检查 Repeat_state 列是否已经存在
    ->     IF NOT EXISTS (
    ->         SELECT *
    ->         FROM information_schema.COLUMNS
    ->         WHERE TABLE_NAME = 'Watch'
    ->         AND COLUMN_NAME = 'Repeat_state'
    ->     ) THEN
    ->         -- 如果列不存在，则添加列
    ->         ALTER TABLE Watch
    ->         ADD Repeat_state BOOLEAN DEFAULT FALSE;
    ->     END IF;
    ->
    ->     -- 更新 Watch 表中的 Repeat_state 列
    ->     UPDATE Watch W1
    ->     SET Repeat_state =
    ->         (SELECT IF(COUNT(*) > 1, TRUE, FALSE)
    ->         FROM Watch W2
    ->         WHERE W1.Movie_no = W2.Movie_no
    ->         AND W1.Viewer_no = W2.Viewer_no);
    -> end$
Query OK, 0 rows affected (0.066 sec)
```

```
obclient(root@sys)[MOVIE]> delimiter ;
obclient(root@sys)[MOVIE]> call update_state();
Query OK, 8 rows affected (0.308 sec)
```

11.在 8-10 题的基础上，查询没有重复观看过电影的观众信息，输出观众姓名和编号

```
obclient(root@sys)[MOVIE]> SELECT Viewer.Viewer_name, Viewer.Viewer_no
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no
    -> WHERE Repeat_state = 'F';
+-------------+-----------+
| Viewer_name | Viewer_no |
+-------------+-----------+
| 李明        | V001      |
| 王红        | V002      |
| 王红        | V002      |
| 张磊        | V003      |
| 赵颖        | V004      |
| 孙杨        | V005      |
+-------------+-----------+
6 rows in set (0.025 sec)
```

12. (*)修改电影表，在 Movie_name 列上增加唯一性索引 Movie_name_index，并按 Movie_name 升序排列。

```
obclient(root@sys)[MOVIE]> ALTER TABLE Movie
    -> ADD UNIQUE INDEX Movie_name_index (Movie_name);
Query OK, 0 rows affected (0.556 sec)
```

出现的问题：

1.使用 orderby 进行排序的时候出现报错

```
obclient(root@sys)[MOVIE]> SELECT Viewer.Viewer_no
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no join Movie on Watch.Movie_no = Movie.Movie_no
    -> WHERE Movie_name = '星际穿越'
    -> EXCEPT
    -> (SELECT Viewer.Viewer_no
    -> FROM Viewer join Watch on Viewer.Viewer_no = Watch.Viewer_no join Movie on Watch.Movie_no = Movie.Movie_no
    -> WHERE Movie_name = '盗梦空间')
    -> ORDER BY Viewer.Viewer_no;
ERROR 1250 (42000): Table 'Viewer' from one of the SELECTs cannot be used in global ORDER clause
[172.17.0.2:2882] [2025-04-30 10:34:05.525735] [YB42AC110002-000633FC5007D0B9-0-0]
```

2.创建过程时发生错误

```
obclient(root@sys)[MOVIE]> CREATE procedure update_state()
    -> begin
    ->
    -> ALTER TABLE Watch
    -> ADD Repeat_state BOOLEAN DEFAULT FALSE ;
    ->
    -> UPDATE Watch W1
    -> SET Repeate_state =
    ->     (SELECT IF(COUNT(*) > 1, TRUE, FALSE)
    ->     FROM Watch W2
    ->     WHERE W1.Movie_no = W2.Movie_no
    ->     AND
    ->     W1.Viewer_no = W2.Viewer_no);
    ->
    -> end$$
ERROR 1304 (42000): PROCEDURE update_state already exists
[172.17.0.2:2882] [2025-04-30 14:46:07.296698] [YB42AC110002-000633FFEBBCBF6A-0-0]
```

解决方案：

1.orderby 后面直接跟属性不指定表名

2.在 Watch 中已经有新属性了，需要删除原来的属性才能避免重复添加