深　圳　大　学

实　　验　　报　　告

课程名称：<u>　　　数据库系统　　　</u>

实验序号：<u>　　　实验 **2**　　　</u>

实验名称：<u>　　**SQL的多表连接查询以及视图**　　</u>

学　　号：<u>　　　**2022155028**　　　</u>

姓　　名：<u>　　　黄亮铭　　　</u>

实验完成日期：<u> **2024** </u>年<u> **10** </u>月<u> **07** </u>日

一、实验目的：

1、 了解 DBMS 系统的功能、软件组成；

2、掌握利用 SQL 语句定义、和简单操纵数据库的方法。

二、实验要求：

1、在课外安装相关软件并浏览软件自带的帮助文件和功能菜单，了解 DBMS 的功能、结构；

2、创建一个有两个关系表的数据库；（建议采用 ORACLE ISQLPLUS）

3、数据库、关系表定义；

4、学习定义关系表的约束(主键、外键、自定义)；

5、了解 SQL 的数据定义功能；

6、了解 SQL 的操纵基本功能；

8、了解视图的概念；

三、实验设备：

计算机、数据库管理系统如 MYSQL,DB2，Oracle 等软件。

四、实验内容

1、使用 SQL DDL 语句**建立关系数据库模式，并用 DML 数据**如下；

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-Dec-90	13750		20
7499	ALLEN	SALESMAN	7698	20-FEB-89	19000	6400	30
7521	WARD	SALESMAN	7698	22-FEB-93	18500	4250	30
7566	JONES	MANAGER	7839	02-APR-89	26850		20
7654	MARTIN	SALESMAN	7698	28-SEP-97	15675	3500	30
7698	BLAKE	MANAGER	7839	01-MAY-90	24000		30
7782	CLARK	MANAGER	7839	09-JUN-88	27500		10
7788	SCOTT	ANALYST	7566	19-APR-87	19500		20
7839	KING	PRESIDENT		17-NOV-83	82500		10
7844	TURNER	SALESMAN	7698	08-SEP-92	18500	6250	30
7876	ADAMS	CLERK	7788	23-MAY-96	11900		20
7900	JAMES	CLERK	7698	03-DEC-95	12500		30
7902	FORD	ANALYST	7566	03-DEC-91	21500		20
7934	MILLER	CLERK	7782	23-JAN-95	13250		10
3258	GREEN	SALESMAN	4422	24-Jul-95	18500	2750	50
4422	STEVENS	MANAGER	7839	14-Jan-94	24750		50

6548	BARNES	CLERK	4422	16-Jan-95	11950	50

DEPT+(学生自己的学号):

DEPTNO	DNAME	LOC
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	STAFFORD
50	MARKETING	LUTON

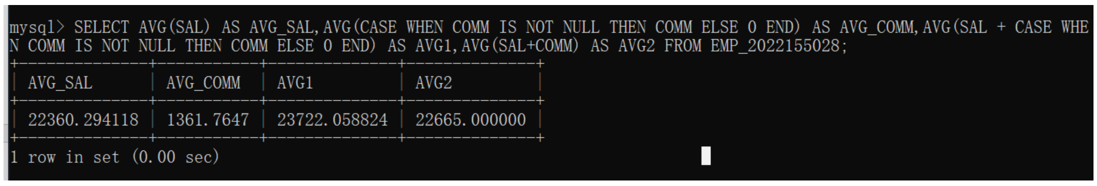
以下为学生实验填写部分：

1.．完成实验指导书的以下题目.（要有题目语句和运行结果截屏）

Exercises3 第 7,8 题

7. List the average salary of employees that receive a salary, the average commission of employees that receive a commission, the average salary plus commission of only those employees that receive a commission and the average salary plus commission of all employees including those that do not receive a commission. (single statement)
输入命令：

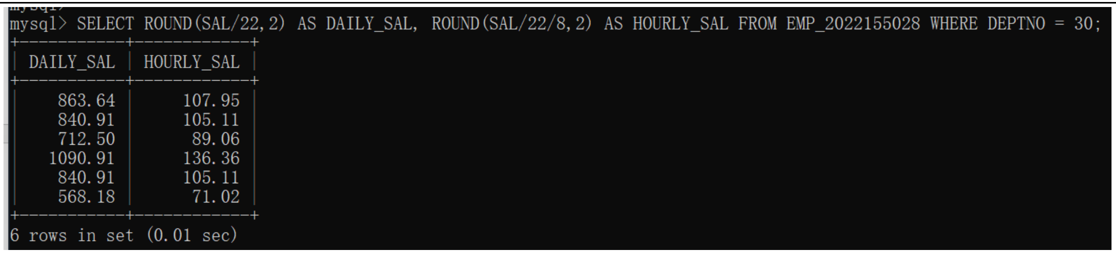
```
SELECT AVG(SAL) AS AVG_SAL,AVG(CASE WHEN COMM IS NOT NULL THEN COMM ELSE 0 END) AS AVG_COMM,AVG(SAL + CASE WHEN COMM IS NOT NULL THEN COMM ELSE 0 END) AS AVG1,AVG(SAL+COMM) AS AVG2 FROM EMP_2022155028;
```



```
mysql> SELECT AVG(SAL) AS AVG_SAL,AVG(CASE WHEN COMM IS NOT NULL THEN COMM ELSE 0 END) AS AVG_COMM,AVG(SAL + CASE WHE
N COMM IS NOT NULL THEN COMM ELSE 0 END) AS AVG1,AVG(SAL+COMM) AS AVG2 FROM EMP_2022155028;
+--------------+-----------+--------------+--------------+
| AVG_SAL      | AVG_COMM  | AVG1         | AVG2         |
+--------------+-----------+--------------+--------------+
| 22360.294118 | 1361.7647 | 23722.058824 | 22665.000000 |
+--------------+-----------+--------------+--------------+
1 row in set (0.00 sec)
```

8. Compute the daily and hourly salary for employees in department 30, round to the nearest penny. Assume there are 22 working days in a month and 8 working hours in a day.

```
SELECT ROUND(SAL/22,2) AS DAILY_SAL, ROUND(SAL/22/8,2) AS HOURLY_SAL FROM EMP_2022155028 WHERE DEPTNO = 30;
```

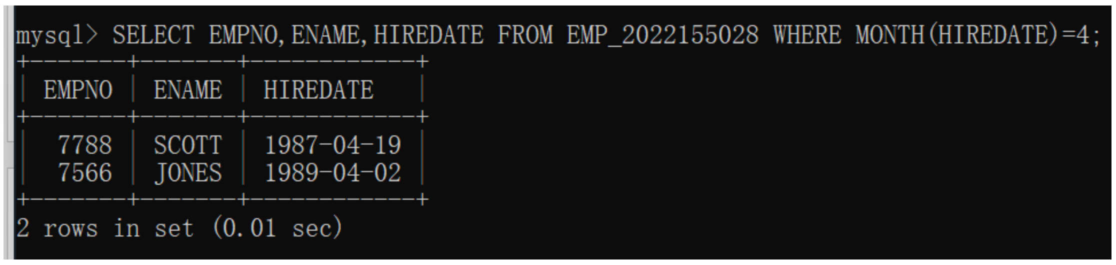


```
mysql> SELECT ROUND(SAL/22,2) AS DAILY_SAL, ROUND(SAL/22/8,2) AS HOURLY_SAL FROM EMP_2022155028 WHERE DEPTNO = 30;
+-----------+------------+
| DAILY_SAL | HOURLY_SAL |
+-----------+------------+
|    863.64 |     107.95 |
|    840.91 |     105.11 |
|    712.50 |      89.06 |
|   1090.91 |     136.36 |
|    840.91 |     105.11 |
|    568.18 |      71.02 |
+-----------+------------+
6 rows in set (0.01 sec)
```

Exercises4 第 3,6,7,8 题

3. Which employees were hired in April?

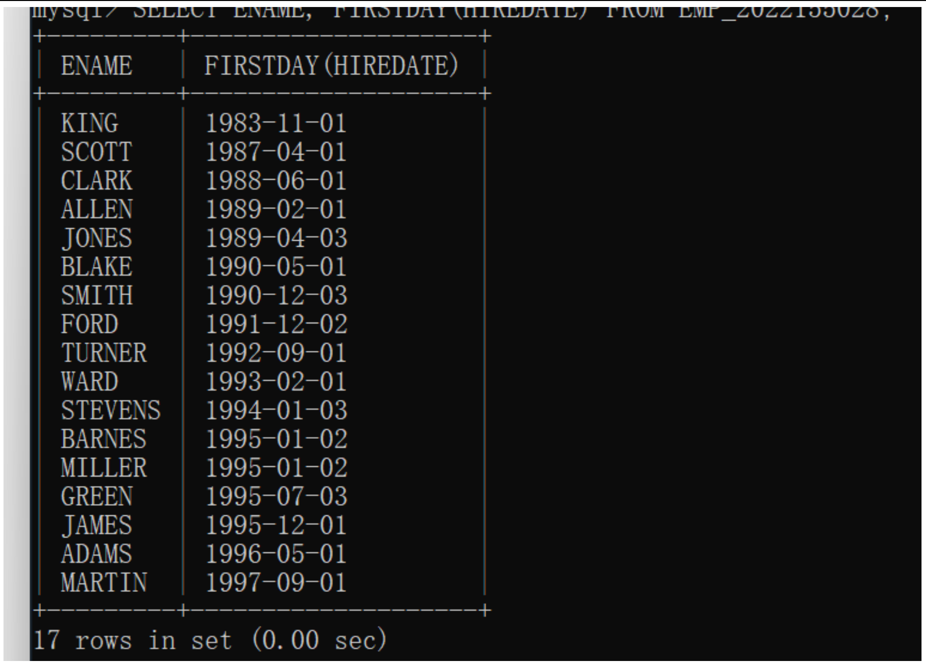
```
SELECT EMPNO,ENAME,HIREDATE FROM EMP_2022155028 WHERE MONTH(HIREDATE)=4;
```



```
mysql> SELECT EMPNO,ENAME,HIREDATE FROM EMP_2022155028 WHERE MONTH(HIREDATE)=4;
+-------+-------+------------+
| EMPNO | ENAME | HIREDATE   |
+-------+-------+------------+
|  7788 | SCOTT | 1987-04-19 |
|  7566 | JONES | 1989-04-02 |
+-------+-------+------------+
2 rows in set (0.01 sec)
```

6．Show the weekday of the first day of the month in which each employee was hired. (plus their names)

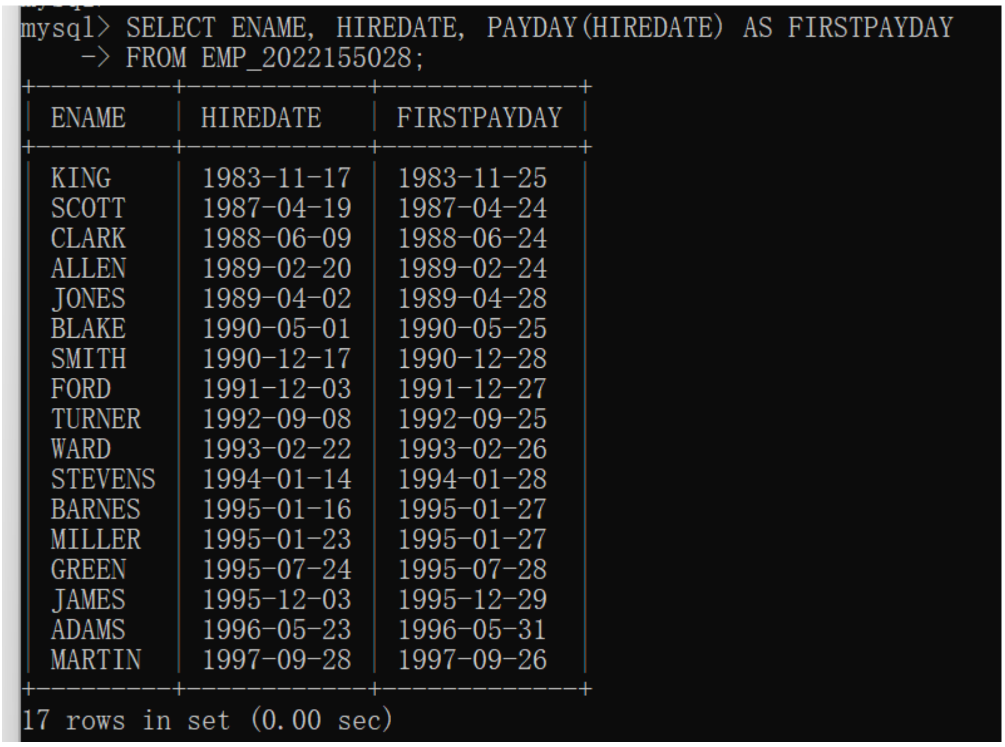
```
set global log_bin_trust_function_creators=TRUE;
DELIMITER $$
DROP FUNCTION IF EXISTS FIRSTDAY $$
CREATE FUNCTION FIRSTDAY(d DATE) RETURNS DATE
BEGIN
DECLARE first DATE;
SET first = DATE(LAST_DAY(d – INTERVAL 1 MONTH) + INTERVAL 1 DAY);
WHILE WEEKDAY(first) > 4 DO
    SET first = DATE(first + INTERVAL 1 DAY);
END WHILE;
RETURN first;
END $$
DELIMITER ;
SELECT ENAME, FIRSTDAY(HIREDATE) FROM EMP_2022155028;
```



```
mysql> SELECT ENAME, FIRSTDAY(HIREDATE) FROM EMP_2022155028;
+---------+--------------------+
| ENAME   | FIRSTDAY(HIREDATE) |
+---------+--------------------+
| KING    | 1983-11-01         |
| SCOTT   | 1987-04-01         |
| CLARK   | 1988-06-01         |
| ALLEN   | 1989-02-01         |
| JONES   | 1989-04-03         |
| BLAKE   | 1990-05-01         |
| SMITH   | 1990-12-03         |
| FORD    | 1991-12-02         |
| TURNER  | 1992-09-01         |
| WARD    | 1993-02-01         |
| STEVENS | 1994-01-03         |
| BARNES  | 1995-01-02         |
| MILLER  | 1995-01-02         |
| GREEN   | 1995-07-03         |
| JAMES   | 1995-12-01         |
| ADAMS   | 1996-05-01         |
| MARTIN  | 1997-09-01         |
+---------+--------------------+
17 rows in set (0.00 sec)
```

7. Show details of employee hiredates and the date of their first payday. (Paydays occur on the last Friday of each month) (plus their names)

```
set global log_bin_trust_function_creators=TRUE;
DELIMITER $$
DROP FUNCTION IF EXISTS PAYDAY $$
CREATE FUNCTION PAYDAY (d DATE) RETURNS DATE
     BEGIN
    DECLARE last DATE;
    SET last=LAST_DAY(d);
    WHILE WEEKDAY(last)!=4 DO
         SET last=DATE(last - INTERVAL 1 DAY);
    END WHILE;
    RETURN last;
    END $$
DELIMITER ;
SELECT ENAME, HIREDATE, PAYDAY(HIREDATE) AS FIRSTPAYDAY
FROM EMP_2022155028;
```

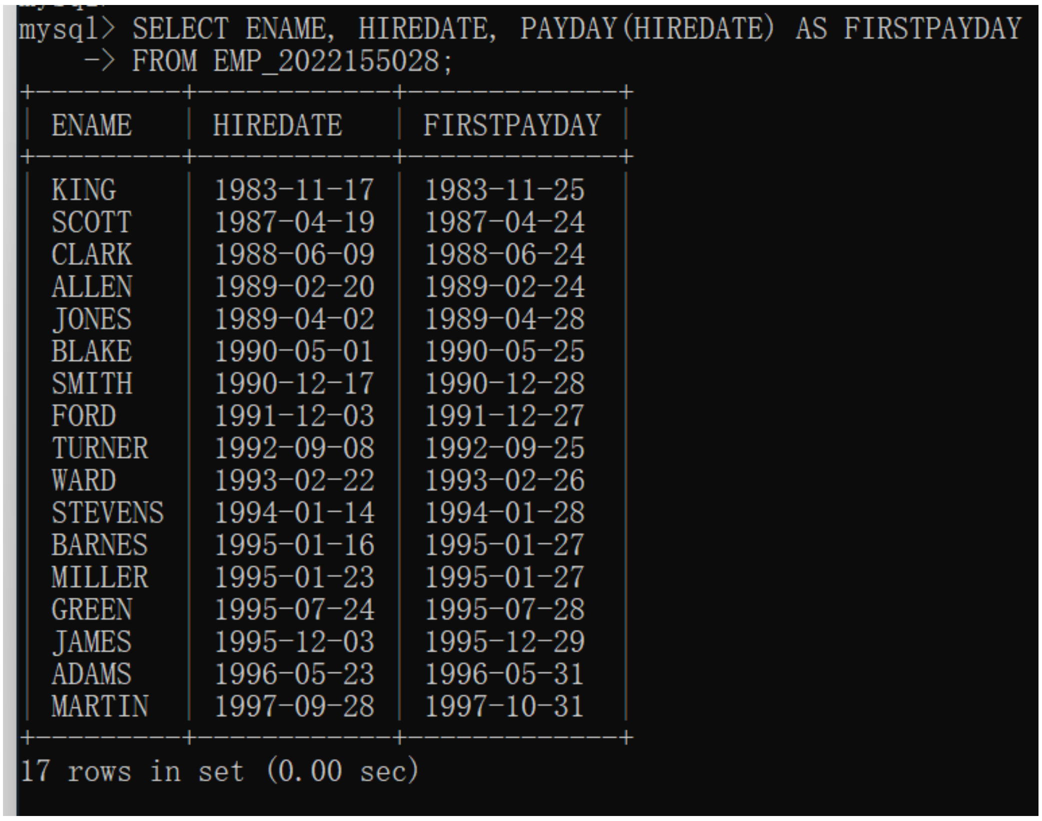


```
mysql> SELECT ENAME, HIREDATE, PAYDAY(HIREDATE) AS FIRSTPAYDAY
    -> FROM EMP_2022155028;
+---------+------------+-------------+
| ENAME   | HIREDATE   | FIRSTPAYDAY |
+---------+------------+-------------+
| KING    | 1983-11-17 | 1983-11-25  |
| SCOTT   | 1987-04-19 | 1987-04-24  |
| CLARK   | 1988-06-09 | 1988-06-24  |
| ALLEN   | 1989-02-20 | 1989-02-24  |
| JONES   | 1989-04-02 | 1989-04-28  |
| BLAKE   | 1990-05-01 | 1990-05-25  |
| SMITH   | 1990-12-17 | 1990-12-28  |
| FORD    | 1991-12-03 | 1991-12-27  |
| TURNER  | 1992-09-08 | 1992-09-25  |
| WARD    | 1993-02-22 | 1993-02-26  |
| STEVENS | 1994-01-14 | 1994-01-28  |
| BARNES  | 1995-01-16 | 1995-01-27  |
| MILLER  | 1995-01-23 | 1995-01-27  |
| GREEN   | 1995-07-24 | 1995-07-28  |
| JAMES   | 1995-12-03 | 1995-12-29  |
| ADAMS   | 1996-05-23 | 1996-05-31  |
| MARTIN  | 1997-09-28 | 1997-09-26  |
+---------+------------+-------------+
17 rows in set (0.00 sec)
```

8. Refine your answer to 7 such that it works even if an employee is hired after the last Friday of the month (cf Martin)

```
set global log_bin_trust_function_creators=TRUE;
DELIMITER $$
DROP FUNCTION IF EXISTS PAYDAY $$
CREATE FUNCTION PAYDAY (d DATE) RETURNS DATE
     BEGIN
    DECLARE last DATE;
    SET last=LAST_DAY(d);
    WHILE WEEKDAY(last)!=4 DO
        SET last=DATE(last - INTERVAL 1 DAY);
    END WHILE;
    IF DAY(d) > DAY(last)
    THEN
        SET last = LAST_DAY(DATE(d + INTERVAL 1 MONTH));
        WHILE WEEKDAY(last)!=4 DO
            SET last=DATE(last - INTERVAL 1 DAY);
        END WHILE;
     END IF;
    RETURN last;
    END $$
DELIMITER ;

SELECT ENAME, HIREDATE, PAYDAY(HIREDATE) AS FIRSTPAYDAY
FROM EMP_2022155028;
```

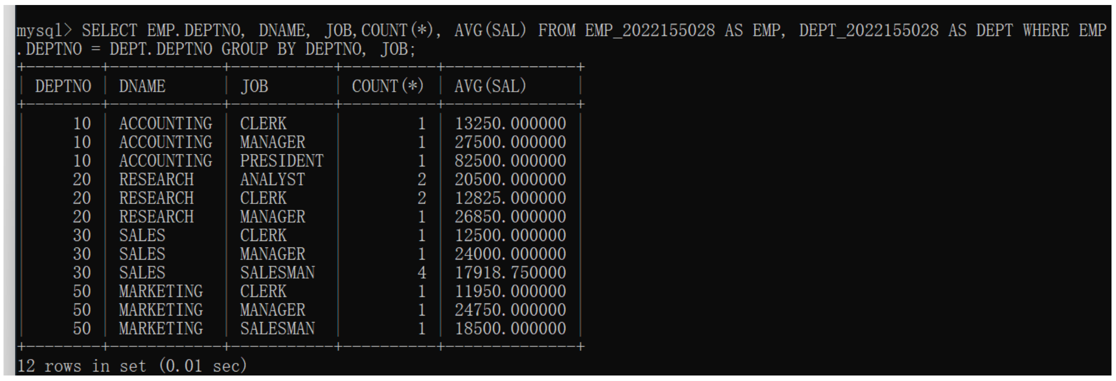


```
mysql> SELECT ENAME, HIREDATE, PAYDAY(HIREDATE) AS FIRSTPAYDAY
    -> FROM EMP_2022155028;
+---------+------------+-------------+
| ENAME   | HIREDATE   | FIRSTPAYDAY |
+---------+------------+-------------+
| KING    | 1983-11-17 | 1983-11-25  |
| SCOTT   | 1987-04-19 | 1987-04-24  |
| CLARK   | 1988-06-09 | 1988-06-24  |
| ALLEN   | 1989-02-20 | 1989-02-24  |
| JONES   | 1989-04-02 | 1989-04-28  |
| BLAKE   | 1990-05-01 | 1990-05-25  |
| SMITH   | 1990-12-17 | 1990-12-28  |
| FORD    | 1991-12-03 | 1991-12-27  |
| TURNER  | 1992-09-08 | 1992-09-25  |
| WARD    | 1993-02-22 | 1993-02-26  |
| STEVENS | 1994-01-14 | 1994-01-28  |
| BARNES  | 1995-01-16 | 1995-01-27  |
| MILLER  | 1995-01-23 | 1995-01-27  |
| GREEN   | 1995-07-24 | 1995-07-28  |
| JAMES   | 1995-12-03 | 1995-12-29  |
| ADAMS   | 1996-05-23 | 1996-05-31  |
| MARTIN  | 1997-09-28 | 1997-10-31  |
+---------+------------+-------------+
17 rows in set (0.00 sec)
```

Exercises,5 第 2,6 题

2. Divide all employees into groups by department and by job within department. Count the employees in each group and compute each group's average annual salary.

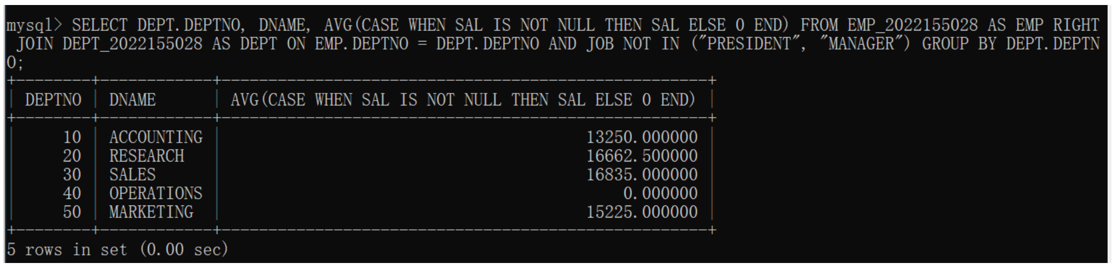
```
SELECT EMP.DEPTNO, DNAME, JOB,COUNT(*), AVG(SAL) FROM
EMP_2022155028 AS EMP, DEPT_2022155028 AS DEPT WHERE EMP.DEPTNO
= DEPT.DEPTNO GROUP BY DEPTNO, JOB;
```



```
mysql> SELECT EMP.DEPTNO, DNAME, JOB,COUNT(*), AVG(SAL) FROM EMP_2022155028 AS EMP, DEPT_2022155028 AS DEPT WHERE EMP
.DEPTNO = DEPT.DEPTNO GROUP BY DEPTNO, JOB;
+--------+------------+-----------+----------+--------------+
| DEPTNO | DNAME      | JOB       | COUNT(*) | AVG(SAL)     |
+--------+------------+-----------+----------+--------------+
|     10 | ACCOUNTING | CLERK     |        1 | 13250.000000 |
|     10 | ACCOUNTING | MANAGER   |        1 | 27500.000000 |
|     10 | ACCOUNTING | PRESIDENT |        1 | 82500.000000 |
|     20 | RESEARCH   | ANALYST   |        2 | 20500.000000 |
|     20 | RESEARCH   | CLERK     |        2 | 12825.000000 |
|     20 | RESEARCH   | MANAGER   |        1 | 26850.000000 |
|     30 | SALES      | CLERK     |        1 | 12500.000000 |
|     30 | SALES      | MANAGER   |        1 | 24000.000000 |
|     30 | SALES      | SALESMAN  |        4 | 17918.750000 |
|     50 | MARKETING  | CLERK     |        1 | 11950.000000 |
|     50 | MARKETING  | MANAGER   |        1 | 24750.000000 |
|     50 | MARKETING  | SALESMAN  |        1 | 18500.000000 |
+--------+------------+-----------+----------+--------------+
12 rows in set (0.01 sec)
```

6. Find each department's average annual salary for all its employees except the managers and the president.

SELECT DEPT.DEPTNO, DNAME, AVG(CASE WHEN SAL IS NOT NULL THEN SAL ELSE 0 END) FROM EMP_2022155028 AS EMP RIGHT JOIN DEPT_2022155028 AS DEPT ON EMP.DEPTNO = DEPT.DEPTNO AND JOB NOT IN (“PRESIDENT”, “MANAGER”) GROUP BY DEPT.DEPTNO;

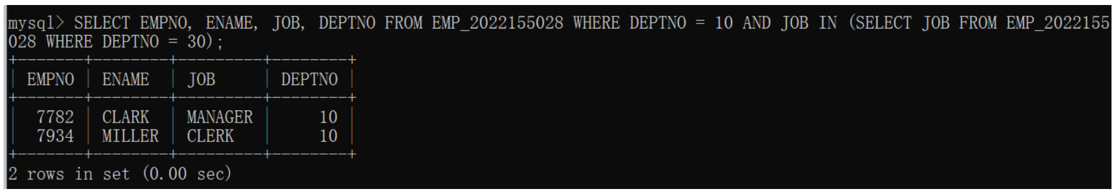


```
mysql> SELECT DEPT.DEPTNO, DNAME, AVG(CASE WHEN SAL IS NOT NULL THEN SAL ELSE 0 END) FROM EMP_2022155028 AS EMP RIGHT
 JOIN DEPT_2022155028 AS DEPT ON EMP.DEPTNO = DEPT.DEPTNO AND JOB NOT IN ("PRESIDENT", "MANAGER") GROUP BY DEPT.DEPTN
O;
+--------+------------+-----------------------------------------------+
| DEPTNO | DNAME      | AVG(CASE WHEN SAL IS NOT NULL THEN SAL ELSE 0 END) |
+--------+------------+-----------------------------------------------+
|     10 | ACCOUNTING |                                  13250.000000 |
|     20 | RESEARCH   |                                  16662.500000 |
|     30 | SALES      |                                  16835.000000 |
|     40 | OPERATIONS |                                      0.000000 |
|     50 | MARKETING  |                                  15225.000000 |
+--------+------------+-----------------------------------------------+
5 rows in set (0.00 sec)
```

Exercises6 第 2,4,5,6,7 题

2. Find all the employees in Department 10 that have a job that is the same as anyone in department 30.

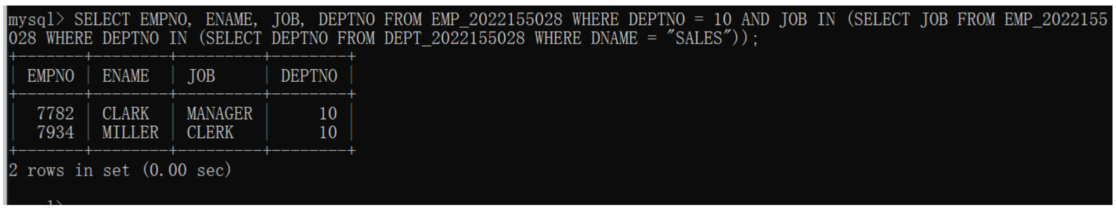
SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP_2022155028 WHERE DEPTNO = 10 AND JOB IN (SELECT JOB FROM EMP_2022155028 WHERE DEPTNO = 30);



```
mysql> SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP_2022155028 WHERE DEPTNO = 10 AND JOB IN (SELECT JOB FROM EMP_2022155
028 WHERE DEPTNO = 30);
+-------+--------+---------+--------+
| EMPNO | ENAME  | JOB     | DEPTNO |
+-------+--------+---------+--------+
|  7782 | CLARK  | MANAGER |     10 |
|  7934 | MILLER | CLERK   |     10 |
+-------+--------+---------+--------+
2 rows in set (0.00 sec)
```

4. Find all employees in department 10 that have a job that is the same as anyone in the Sales department

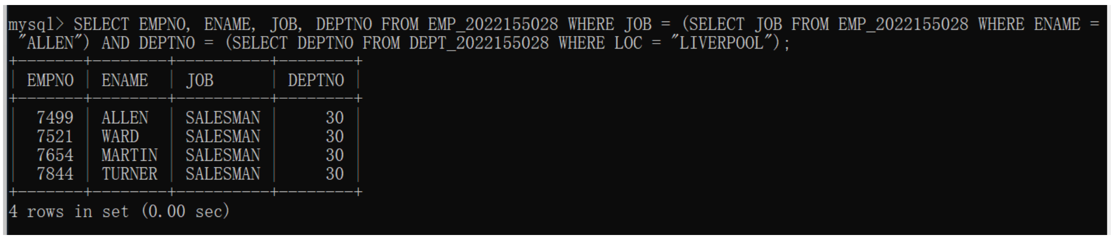
SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP_2022155028 WHERE DEPTNO = 10 AND JOB IN (SELECT JOB FROM EMP_2022155028 WHERE DEPTNO IN (SELECT DEPTNO FROM DEPT_2022155028 WHERE DNAME = “SALES”));



```
mysql> SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP_2022155028 WHERE DEPTNO = 10 AND JOB IN (SELECT JOB FROM EMP_2022155
028 WHERE DEPTNO IN (SELECT DEPTNO FROM DEPT_2022155028 WHERE DNAME = "SALES"));
+-------+--------+---------+--------+
| EMPNO | ENAME  | JOB     | DEPTNO |
+-------+--------+---------+--------+
|  7782 | CLARK  | MANAGER |     10 |
|  7934 | MILLER | CLERK   |     10 |
+-------+--------+---------+--------+
2 rows in set (0.00 sec)
```

5. Find the employees located in Liverpool who have the same job as Allen. Return the results in alphabetical order by employee name.

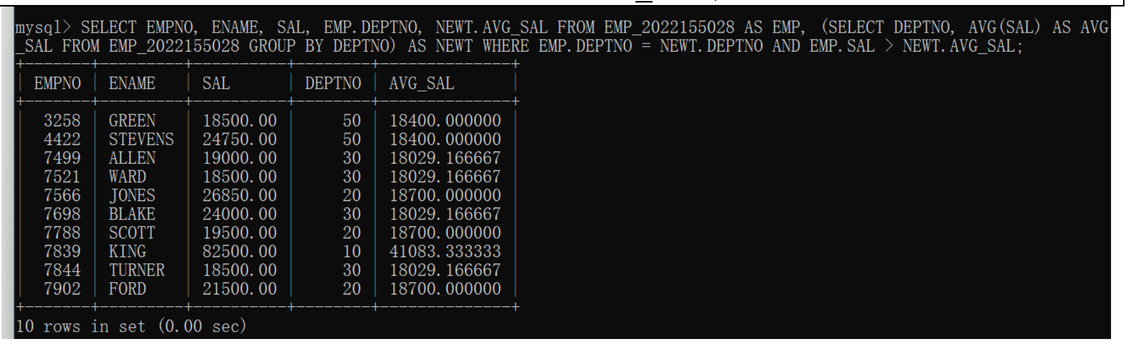
```
SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP_2022155028 WHERE JOB = (SELECT JOB FROM EMP_2022155028 WHERE ENAME = “ALLEN”) AND DEPTNO = (SELECT DEPTNO FROM DEPT_2022155028 WHERE LOC = “LIVERPOOL”);
```



```
mysql> SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP_2022155028 WHERE JOB = (SELECT JOB FROM EMP_2022155028 WHERE ENAME =
 "ALLEN") AND DEPTNO = (SELECT DEPTNO FROM DEPT_2022155028 WHERE LOC = "LIVERPOOL");
+-------+--------+----------+--------+
| EMPNO | ENAME  | JOB      | DEPTNO |
+-------+--------+----------+--------+
|  7499 | ALLEN  | SALESMAN |     30 |
|  7521 | WARD   | SALESMAN |     30 |
|  7654 | MARTIN | SALESMAN |     30 |
|  7844 | TURNER | SALESMAN |     30 |
+-------+--------+----------+--------+
4 rows in set (0.00 sec)
```

6. Find all the employees that earn more than the average salary of employees in their department.

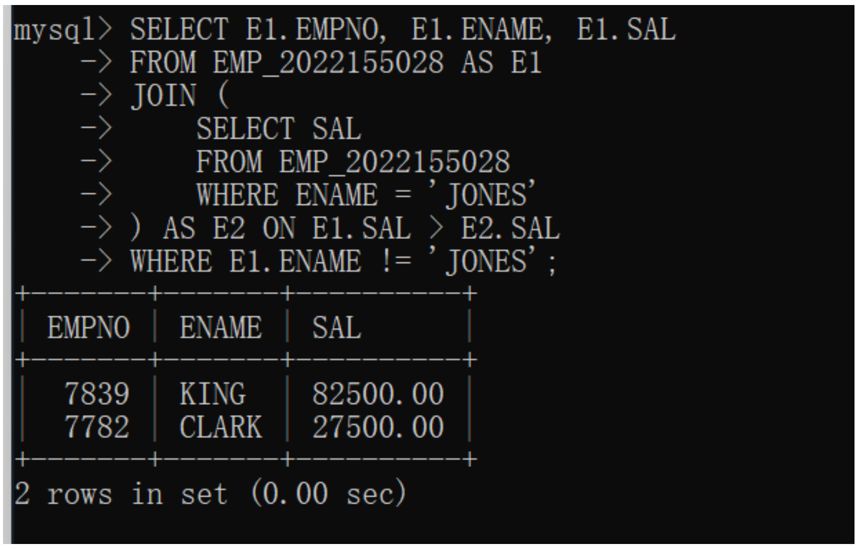
```
SELECT EMPNO, ENAME, SAL, EMP.DEPTNO, NEWT.AVG_SAL FROM EMP_2022155028 AS EMP, (SELECT DEPTNO, AVG(SAL) AS AVG_SAL FROM EMP_2022155028 GROUP BY DEPTNO) AS NEWT WHERE EMP.DEPTNO = NEWT.DEPTNO AND EMP.SAL > NEWT.AVG_SAL;
```



```
mysql> SELECT EMPNO, ENAME, SAL, EMP.DEPTNO, NEWT.AVG_SAL FROM EMP_2022155028 AS EMP, (SELECT DEPTNO, AVG(SAL) AS AVG
_SAL FROM EMP_2022155028 GROUP BY DEPTNO) AS NEWT WHERE EMP.DEPTNO = NEWT.DEPTNO AND EMP.SAL > NEWT.AVG_SAL;
+-------+---------+----------+--------+--------------+
| EMPNO | ENAME   | SAL      | DEPTNO | AVG_SAL      |
+-------+---------+----------+--------+--------------+
|  3258 | GREEN   | 18500.00 |     50 | 18400.000000 |
|  4422 | STEVENS | 24750.00 |     50 | 18400.000000 |
|  7499 | ALLEN   | 19000.00 |     30 | 18029.166667 |
|  7521 | WARD    | 18500.00 |     30 | 18029.166667 |
|  7566 | JONES   | 26850.00 |     20 | 18700.000000 |
|  7698 | BLAKE   | 24000.00 |     30 | 18029.166667 |
|  7788 | SCOTT   | 19500.00 |     20 | 18700.000000 |
|  7839 | KING    | 82500.00 |     10 | 41083.333333 |
|  7844 | TURNER  | 18500.00 |     30 | 18029.166667 |
|  7902 | FORD    | 21500.00 |     20 | 18700.000000 |
+-------+---------+----------+--------+--------------+
10 rows in set (0.00 sec)
```

7. Find all the employees that earn more than JONES, using temporary labels to abbreviate table names.

```
SELECT E1.EMPNO, E1.ENAME, E1.SAL
FROM EMP_2022155028 AS E1
JOIN (
    SELECT SAL
    FROM EMP_2022155028
    WHERE ENAME = 'JONES'
) AS E2 ON E1.SAL > E2.SAL
WHERE E1.ENAME != 'JONES';
```



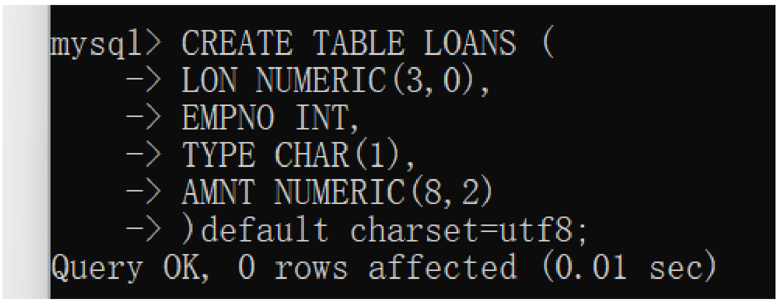
```
mysql> SELECT E1.EMPNO, E1.ENAME, E1.SAL
    -> FROM EMP_2022155028 AS E1
    -> JOIN (
    ->     SELECT SAL
    ->     FROM EMP_2022155028
    ->     WHERE ENAME = 'JONES'
    -> ) AS E2 ON E1.SAL > E2.SAL
    -> WHERE E1.ENAME != 'JONES';
+-------+-------+----------+
| EMPNO | ENAME | SAL      |
+-------+-------+----------+
|  7839 | KING  | 82500.00 |
|  7782 | CLARK | 27500.00 |
+-------+-------+----------+
2 rows in set (0.00 sec)
```

Exercises7 第 2,5,7,8,9 题

2. Insert the following data

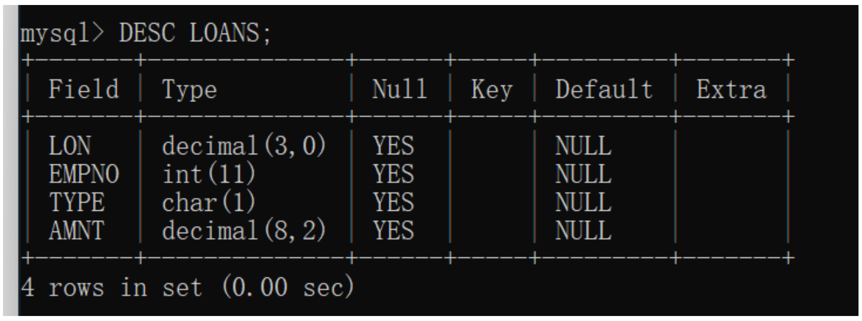
LNO	EMPNO	TYPE	AMNT
23	7499	M	20000.00
42	7499	C	2000.00
65	7844	M	3564.00

```
CREATE TABLE LOANS (
LON NUMERIC(3,0),
EMPNO INT,
TYPE CHAR(1),
AMNT NUMERIC(8,2)
)default charset=utf8;
```



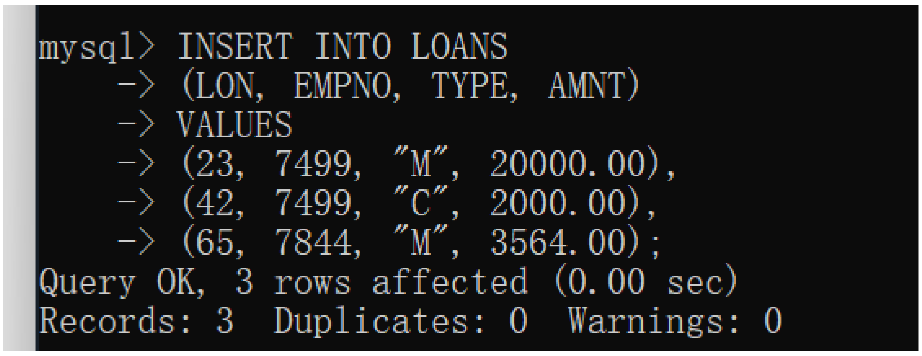
```
mysql> CREATE TABLE LOANS (
    -> LON NUMERIC(3,0),
    -> EMPNO INT,
    -> TYPE CHAR(1),
    -> AMNT NUMERIC(8,2)
    -> )default charset=utf8;
Query OK, 0 rows affected (0.01 sec)
```

输入命令：*DESC LOANS;*查看创建表的结果。



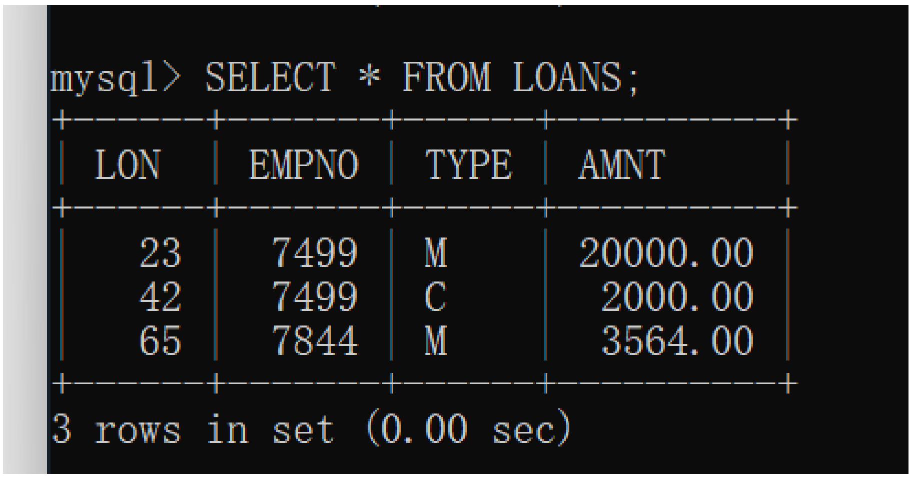
```
mysql> DESC LOANS;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| LON   | decimal(3,0) | YES  |     | NULL    |       |
| EMPNO | int(11)      | YES  |     | NULL    |       |
| TYPE  | char(1)      | YES  |     | NULL    |       |
| AMNT  | decimal(8,2) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
INSERT INTO LOANS
(LON, EMPNO, TYPE, AMNT)
VALUES
(23, 7499, “M”, 20000.00),
(42, 7499, “C”, 2000.00),
(65, 7844, “M”, 3564.00);
```



```
mysql> INSERT INTO LOANS
    -> (LON, EMPNO, TYPE, AMNT)
    -> VALUES
    -> (23, 7499, "M", 20000.00),
    -> (42, 7499, "C", 2000.00),
    -> (65, 7844, "M", 3564.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

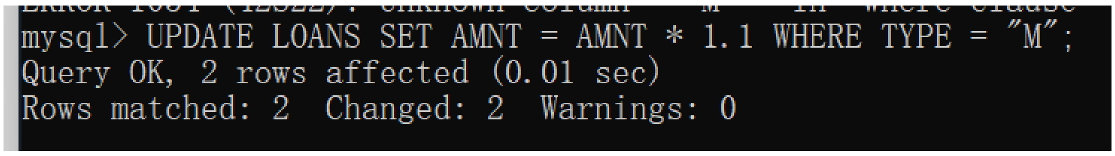
输入命令：*SELECT * FROM LOANS;*查看插入结果。



```
mysql> SELECT * FROM LOANS;
+------+--------+------+----------+
| LON  | EMPNO  | TYPE | AMNT     |
+------+--------+------+----------+
|   23 |   7499 | M    | 20000.00 |
|   42 |   7499 | C    |  2000.00 |
|   65 |   7844 | M    |  3564.00 |
+------+--------+------+----------+
3 rows in set (0.00 sec)
```

5．Add 10% interest to all M type loans

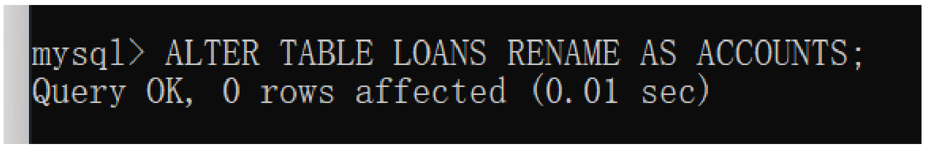
```
UPDATE LOANS SET AMNT = AMNT * 1.1 WHERE TYPE = “M”;
```



```
mysql> UPDATE LOANS SET AMNT = AMNT * 1.1 WHERE TYPE = "M";
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

7. Change the name of loans table to accounts

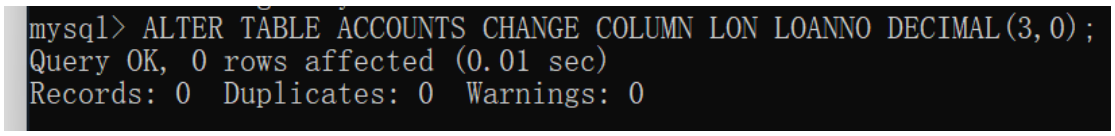
```
ALTER TABLE LOANS RENAME AS ACCOUNTS;
```



```
mysql> ALTER TABLE LOANS RENAME AS ACCOUNTS;
Query OK, 0 rows affected (0.01 sec)
```

8. Change the name of column LNO to LOANNO

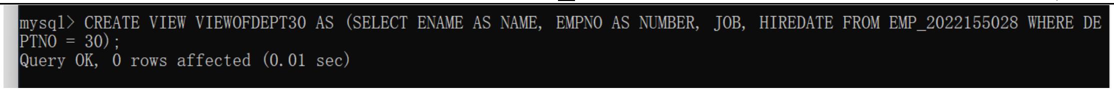
```
ALTER TABLE ACCOUNTS CHANGE COLUMN LON LOANNO
DECIMAL(3,0);
```



```
mysql> ALTER TABLE ACCOUNTS CHANGE COLUMN LON LOANNO DECIMAL(3,0);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

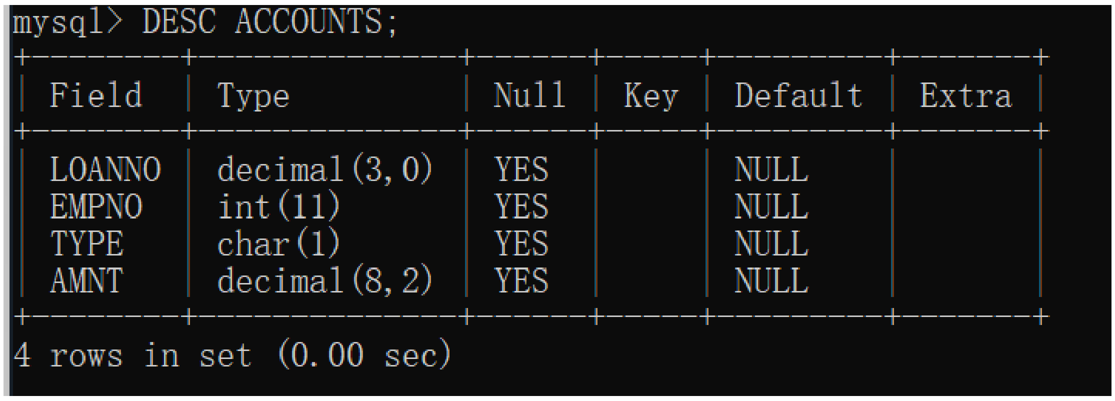
9. Create a view for use by personnel in department 30 showing employee name, number, job and hiredate

```
CREATE VIEW VIEWOFDEPT30 AS (SELECT ENAME AS NAME, EMPNO AS NUMBER, JOB, HIREDATE FROM EMP_2022155028 WHERE DEPTNO = 30);
```



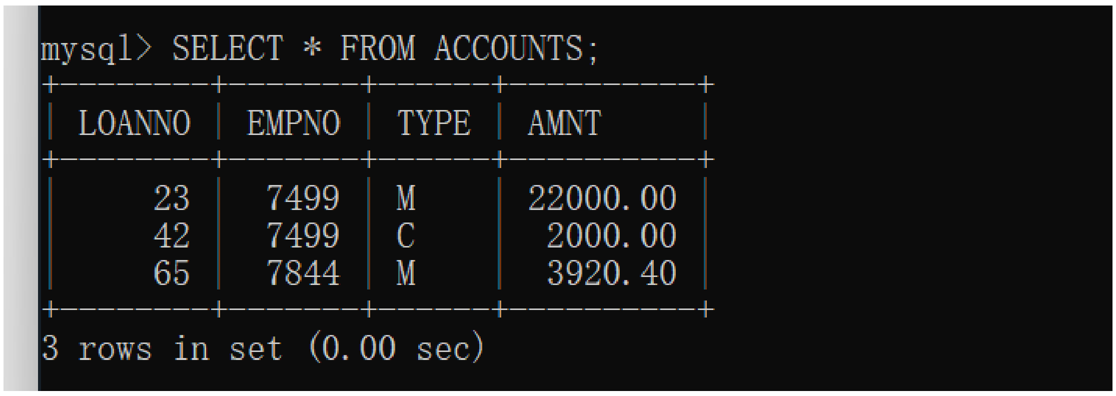
```
mysql> CREATE VIEW VIEWOFDEPT30 AS (SELECT ENAME AS NAME, EMPNO AS NUMBER, JOB, HIREDATE FROM EMP_2022155028 WHERE DE
PTNO = 30);
Query OK, 0 rows affected (0.01 sec)
```

经过 Exercises7 的创建表、添加数据以及修改之后，表的结构和数据内容如下图所示。



```
mysql> DESC ACCOUNTS;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| LOANNO | decimal(3,0) | YES  |     | NULL    |       |
| EMPNO  | int(11)      | YES  |     | NULL    |       |
| TYPE   | char(1)      | YES  |     | NULL    |       |
| AMNT   | decimal(8,2) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

表结构



```
mysql> SELECT * FROM ACCOUNTS;
+--------+-------+------+----------+
| LOANNO | EMPNO | TYPE | AMNT     |
+--------+-------+------+----------+
|     23 |  7499 | M    | 22000.00 |
|     42 |  7499 | C    |  2000.00 |
|     65 |  7844 | M    |  3920.40 |
+--------+-------+------+----------+
3 rows in set (0.00 sec)
```

数据内容

开放性题目

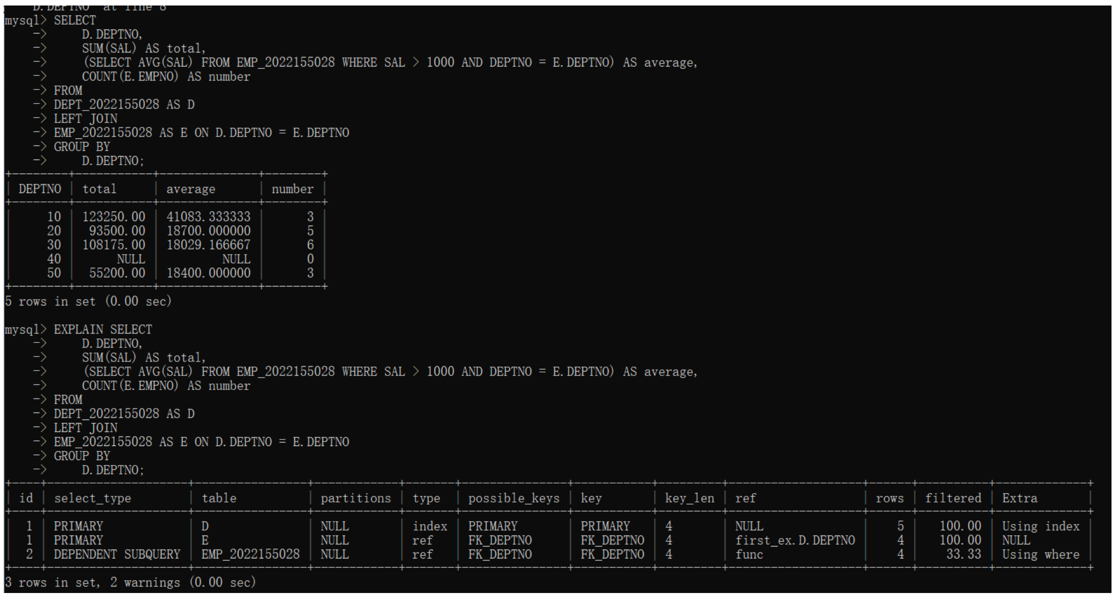
你需要编写一个复杂的查询，分析不同部门的员工总工资、部门中工资高于某个门槛（例如 $1000）的员工平均工资，以及每个部门的员工人数。查询中 employee 表可能会被扫描多次：一次用于计算总工资，另一次用于计算平均高薪员工工资，第三次用于计算员工人数。通过使用子查询或临时表优化查询，可以减少重复扫描，提升查询性能。并使用 EXPLAIN 比较优化前后的查询执行性能。如有可能，请在大数据集上验证查询优化的实际效果。

普通查询的 SQL 语句。

```
SELECT
```

```
    D.DEPTNO,
    SUM(SAL) AS total,
    (SELECT AVG(SAL) FROM EMP_2022155028 WHERE SAL > 1000 AND DEPTNO =
E.DEPTNO) AS average,
    COUNT(E.EMPNO) AS number
FROM
    DEPT_2022155028 AS D
LEFT JOIN
    EMP_2022155028 AS E ON D.DEPTNO = E.DEPTNO
GROUP BY
    D.DEPTNO;
```

普通查询的查询结果以及执行性能如下图所示。



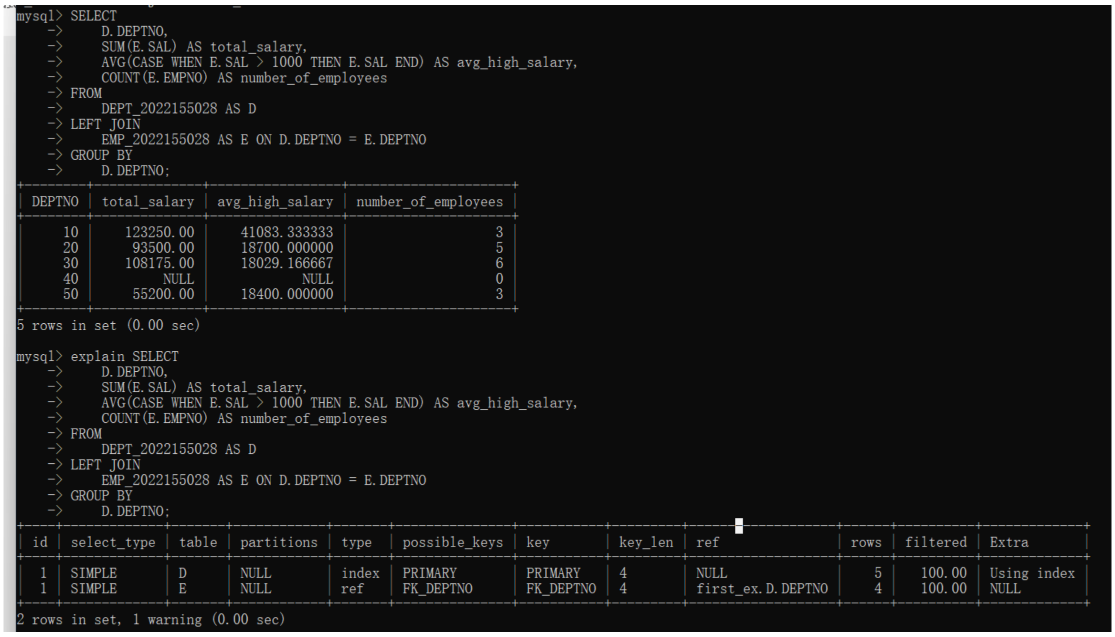
```
    D.DEPTNO  at line 8
mysql> SELECT
    ->     D.DEPTNO,
    ->     SUM(SAL) AS total,
    ->     (SELECT AVG(SAL) FROM EMP_2022155028 WHERE SAL > 1000 AND DEPTNO = E.DEPTNO) AS average,
    ->     COUNT(E.EMPNO) AS number
    -> FROM
    -> DEPT_2022155028 AS D
    -> LEFT JOIN
    -> EMP_2022155028 AS E ON D.DEPTNO = E.DEPTNO
    -> GROUP BY
    ->     D.DEPTNO;
+--------+-----------+--------------+--------+
| DEPTNO | total     | average      | number |
+--------+-----------+--------------+--------+
|     10 | 123250.00 | 41083.333333 |      3 |
|     20 |  93500.00 | 18700.000000 |      5 |
|     30 | 108175.00 | 18029.166667 |      6 |
|     40 |      NULL |         NULL |      0 |
|     50 |  55200.00 | 18400.000000 |      3 |
+--------+-----------+--------------+--------+
5 rows in set (0.00 sec)

mysql> EXPLAIN SELECT
    ->     D.DEPTNO,
    ->     SUM(SAL) AS total,
    ->     (SELECT AVG(SAL) FROM EMP_2022155028 WHERE SAL > 1000 AND DEPTNO = E.DEPTNO) AS average,
    ->     COUNT(E.EMPNO) AS number
    -> FROM
    -> DEPT_2022155028 AS D
    -> LEFT JOIN
    -> EMP_2022155028 AS E ON D.DEPTNO = E.DEPTNO
    -> GROUP BY
    ->     D.DEPTNO;
+----+--------------------+----------------+------------+-------+---------------+------------+---------+--------------------+------+----------+-------------+
| id | select_type        | table          | partitions | type  | possible_keys | key        | key_len | ref                | rows | filtered | Extra       |
+----+--------------------+----------------+------------+-------+---------------+------------+---------+--------------------+------+----------+-------------+
|  1 | PRIMARY            | D              | NULL       | index | PRIMARY       | PRIMARY    | 4       | NULL               |    5 |   100.00 | Using index |
|  1 | PRIMARY            | E              | NULL       | ref   | FK_DEPTNO     | FK_DEPTNO  | 4       | first_ex.D.DEPTNO  |    4 |   100.00 | NULL        |
|  2 | DEPENDENT SUBQUERY | EMP_2022155028 | NULL       | ref   | FK_DEPTNO     | FK_DEPTNO  | 4       | func               |    4 |    33.33 | Using where |
+----+--------------------+----------------+------------+-------+---------------+------------+---------+--------------------+------+----------+-------------+
3 rows in set, 2 warnings (0.00 sec)
```

优化查询的 SQL 语句如下图所示。主要作用为减少了一次子查询。

```
SELECT
    D.DEPTNO,
    SUM(E.SAL) AS total_salary,
    AVG(CASE WHEN E.SAL > 1000 THEN E.SAL END) AS avg_high_salary,
    COUNT(E.EMPNO) AS number_of_employees
FROM
    DEPT_2022155028 AS D
LEFT JOIN
    EMP_2022155028 AS E ON D.DEPTNO = E.DEPTNO
GROUP BY
    D.DEPTNO;
```

优化查询结果及执行性能如下图所示。由普通查询和优化查询的执行性能的比较可知：优化查询相比于普通查询少了一次子查询，在大数据的前提下，减少一次子查询可以大大缩短查询时间，优化效果会非常显著。



```
mysql> SELECT
    ->     D. DEPTNO,
    ->     SUM(E. SAL) AS total_salary,
    ->     AVG(CASE WHEN E. SAL > 1000 THEN E. SAL END) AS avg_high_salary,
    ->     COUNT(E. EMPNO) AS number_of_employees
    -> FROM
    ->     DEPT_2022155028 AS D
    -> LEFT JOIN
    ->     EMP_2022155028 AS E ON D. DEPTNO = E. DEPTNO
    -> GROUP BY
    ->     D. DEPTNO;
+--------+--------------+-----------------+---------------------+
| DEPTNO | total_salary | avg_high_salary | number_of_employees |
+--------+--------------+-----------------+---------------------+
|     10 |    123250.00 |    41083.333333 |                   3 |
|     20 |     93500.00 |    18700.000000 |                   5 |
|     30 |    108175.00 |    18029.166667 |                   6 |
|     40 |         NULL |            NULL |                   0 |
|     50 |     55200.00 |    18400.000000 |                   3 |
+--------+--------------+-----------------+---------------------+
5 rows in set (0.00 sec)

mysql> explain SELECT
    ->     D. DEPTNO,
    ->     SUM(E. SAL) AS total_salary,
    ->     AVG(CASE WHEN E. SAL > 1000 THEN E. SAL END) AS avg_high_salary,
    ->     COUNT(E. EMPNO) AS number_of_employees
    -> FROM
    ->     DEPT_2022155028 AS D
    -> LEFT JOIN
    ->     EMP_2022155028 AS E ON D. DEPTNO = E. DEPTNO
    -> GROUP BY
    ->     D. DEPTNO;
+----+-------------+-------+------------+-------+---------------+-----------+---------+--------------------+------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref                | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+-----------+---------+--------------------+------+----------+-------------+
|  1 | SIMPLE      | D     | NULL       | index | PRIMARY       | PRIMARY   | 4       | NULL               |    5 |   100.00 | Using index |
|  1 | SIMPLE      | E     | NULL       | ref   | FK_DEPTNO     | FK_DEPTNO | 4       | first_ex.D. DEPTNO |    4 |   100.00 | NULL        |
+----+-------------+-------+------------+-------+---------------+-----------+---------+--------------------+------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```

五. 实验心得

1. 通过本次实验，我学会了如何创建和管理数据库中的表，以及如何通过 SQL 语句进行复杂的数据查询。
2. 通过连接查询，我能够从多个表中提取出有用的信息，这对于数据分析来说是非常有价值的。
3. 视图的创建让我学会了如何简化复杂的查询，使得数据的呈现更加直观和易于理解。

六. 诚信承诺

本人郑重承诺在 SQL 实验的实施的过程中不发生任何不诚信现象，一切不诚信所导致的后果均由本人承担

签名（手签，不得打印）：[illegible]

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：