MySQL学习笔记（Day011：SELECT）

MySQL学习

一. SELECT语法介绍

> SELECT语法官方文档

```
SELECT
-- -------------------------不推荐使用--------------------------
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [MAX_STATEMENT_TIME = N]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
-- ------------------------------------------------------------
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

二. LIMIT 和 ORDER BY

```
mysql> select * from employees limit 1;  -- 从employees中 随机 取出一条数据，结果是不确定的
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |
+--------+------------+------------+-----------+--------+------------+
1 row in set (0.00 sec)

--
-- order by col_name 根据某列的值进行排序
-- asc ： 升序(default)
-- desc： 降序
--

mysql> select * from employees order by emp_no asc  limit 1;   -- 使用order by col_name asc进行升序排序
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |
+--------+------------+------------+-----------+--------+------------+
1 row in set (0.00 sec)

mysql> select * from employees order by  emp_no limit 1;       -- 默认就是升序的
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |  -- 结果和上面一致
+--------+------------+------------+-----------+--------+------------+
1 row in set (0.00 sec)

mysql> select * from employees order by emp_no desc limit 1;    -- desc表示降序
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 499999 | 1958-05-01 | Sachin     | Tsukuda   | M      | 1997-11-30 |  -- 降序显示
+--------+------------+------------+-----------+--------+------------+
-- 通过order by排序后 limit 1 才是确定的
1 row in set (0.00 sec)

mysql> show create table employees\G
*************************** 1. row ***************************
       Table: employees
Create Table: CREATE TABLE `employees` (
  `emp_no` int(11) NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)    -- emp_no 是主键，order by 主键 不会创建临时表的，主键(索引)本身有序
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
1 row in set (0.00 sec)

mysql> select * from employees order by emp_no asc limit 5,5;  -- limit start, offset
                                                               -- 从第5条 开始取，取5条出来
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10006 | 1953-04-20 | Anneke     | Preusig   | F      | 1989-06-02 |
|  10007 | 1957-05-23 | Tzvetan    | Zielinski | F      | 1989-02-10 |
|  10008 | 1958-02-19 | Saniya     | Kalloufi  | M      | 1994-09-15 |
|  10009 | 1952-04-19 | Sumant     | Peac      | F      | 1985-02-18 |
|  10010 | 1963-06-01 | Duangkaew  | Piveteau  | F      | 1989-08-24 |
+--------+------------+------------+-----------+--------+------------+
5 rows in set (0.00 sec)

-- 以上这个语法有一种分页的效果，但是会随着start的增加，性能会下降，因为会扫描表(从 1 到 start)

-- 相对比较推荐的方法
mysql> select * from employees where emp_no > 20000 order by emp_no limit 5;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  20001 | 1962-05-16 | Atreye     | Eppinger  | M      | 1990-04-18 |
|  20002 | 1955-12-25 | Jaber      | Brender   | M      | 1988-01-26 |
|  20003 | 1953-04-11 | Munehiko   | Coors     | F      | 1991-02-07 |
|  20004 | 1952-03-07 | Radoslaw   | Pfau      | M      | 1995-11-24 |
|  20005 | 1956-02-20 | Licheng    | Przulj    | M      | 1992-07-17 |
+--------+------------+------------+-----------+--------+------------+
5 rows in set (0.00 sec)
-- （当然推荐把热数据放cache里,比如Redis）
```

> ORDER BY 是把已经查询好的结果集进行排序

三. WHERE

WHERE 是将查询出来的结果，通过 WHERE 后面的条件(condition)，对结果进行过滤

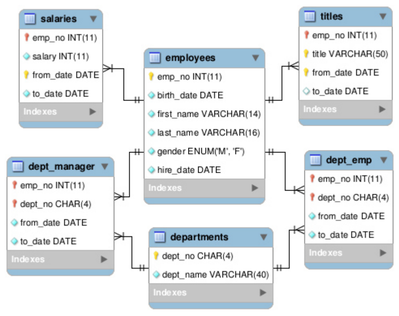
```
mysql> select * from employees  where emp_no > 30000  emp_no limit 4; -- 不加order by的limit是不确定的SQL
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  30001 | 1953-03-27 | Izaskun    | Morton    | M      | 1988-05-21 |
|  30002 | 1960-08-23 | Branimir   | Snedden   | M      | 1998-09-24 |
|  30003 | 1952-11-25 | Takahito   | Vilarrasa | M      | 1990-08-22 |
|  30004 | 1957-11-26 | Lucian     | Penttonen | F      | 1992-10-08 |
+--------+------------+------------+-----------+--------+------------+
4 rows in set (0.00 sec)

mysql> select * from employees  where emp_no > 40000  order by emp_no limit 4;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  40001 | 1956-03-28 | Akemi      | Maliniak  | F      | 1987-08-06 |
|  40002 | 1960-03-15 | Nakhoon    | Badr      | M      | 1990-02-13 |
|  40003 | 1960-01-26 | Jacopo     | Marshall  | F      | 1991-09-30 |
|  40004 | 1955-09-09 | Anneke     | Stiles    | F      | 1986-03-05 |
+--------+------------+------------+-----------+--------+------------+
4 rows in set (0.02 sec)

mysql> select * from employees
    -> where emp_no > 40000
    -> and hire_date > '1991-01-01'   -- 可以用 and 进行 逻辑与
    -> order by emp_no limit 4;
+--------+------------+------------+------------+--------+------------+
| emp_no | birth_date | first_name | last_name  | gender | hire_date  |
+--------+------------+------------+------------+--------+------------+
|  40003 | 1960-01-26 | Jacopo     | Marshall   | F      | 1991-09-30 |
|  40005 | 1961-02-27 | Zsolt      | Fairtlough | F      | 1991-07-08 |
|  40012 | 1955-02-07 | Chinhyun   | Ozeri      | F      | 1995-08-12 |
|  40015 | 1964-10-08 | Ioana      | Lemarechal | M      | 1997-08-07 |
+--------+------------+------------+------------+--------+------------+
4 rows in set (0.00 sec)

mysql> select * from employees
    -> where (emp_no > 40000 and birth_date > '1961-01-01') -- 使用()明确条件的逻辑规则
    ->    or (emp_no > 40000 and hire_date > '1991-01-01')  -- 可以使用 or 做 逻辑或
    -> order by emp_no limit 5;
+--------+------------+------------+------------+--------+------------+
| emp_no | birth_date | first_name | last_name  | gender | hire_date  |
+--------+------------+------------+------------+--------+------------+
|  40003 | 1960-01-26 | Jacopo     | Marshall   | F      | 1991-09-30 |
|  40005 | 1961-02-27 | Zsolt      | Fairtlough | F      | 1991-07-08 |
|  40006 | 1962-11-07 | Basim      | Panienski  | F      | 1986-12-27 |
|  40012 | 1955-02-07 | Chinhyun   | Ozeri      | F      | 1995-08-12 |
|  40015 | 1964-10-08 | Ioana      | Lemarechal | M      | 1997-08-07 |
+--------+------------+------------+------------+--------+------------+
5 rows in set (0.00 sec)
```

四. JOIN



4.1. INNER JOIN

```
--
-- ANSI SQL 89
-- 关联employees表和titles表
-- 要求是 employees的emp_no 等于 titles的emp_no
--
mysql> select * from employees,titles where employees.emp_no = titles.emp_no limit 5;
+--------+------------+------------+-----------+--------+------------+--------+-----------------+------------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  | emp_no | title           | from_date  | to_date    |
+--------+------------+------------+-----------+--------+------------+--------+-----------------+------------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |  10001 | Senior Engineer | 1986-06-26 | 9999-01-01 |
|  10002 | 1964-06-02 | Bezalel    | Simmel    | F      | 1985-11-21 |  10002 | Staff           | 1996-08-03 | 9999-01-01 |
|  10003 | 1959-12-03 | Parto      | Bamford   | M      | 1986-08-28 |  10003 | Senior Engineer | 1995-12-03 | 9999-01-01 |
|  10004 | 1954-05-01 | Chirstian  | Koblick   | M      | 1986-12-01 |  10004 | Engineer        | 1986-12-01 | 1995-12-01 |
|  10004 | 1954-05-01 | Chirstian  | Koblick   | M      | 1986-12-01 |  10004 | Senior Engineer | 1995-12-01 | 9999-01-01 |
+--------+------------+------------+-----------+--------+------------+--------+-----------------+------------+------------+
5 rows in set (0.00 sec)

--
-- 在上面的基础上只显示emp_no，名字，性别和职位名称
--
mysql> select emp_no, concat(last_name,' ', first_name), gender, title
    -> from employees,titles
    -> where employees.emp_no = titles.emp_no limit 5;
ERROR 1052 (23000): Column 'emp_no' in field list is ambiguous  -- 报错了，原因是emp_no两个表都有

mysql> select employees.emp_no, -- 指定了employees
    -> concat(last_name,' ', first_name), gender, title
    -> from employees,titles
    -> where employees.emp_no = titles.emp_no limit 5;
+--------+-----------------------------------+--------+-----------------+
| emp_no | concat(last_name,' ', first_name) | gender | title           |
+--------+-----------------------------------+--------+-----------------+
|  10001 | Facello Georgi                    | M      | Senior Engineer |
|  10002 | Simmel Bezalel                    | F      | Staff           |
|  10003 | Bamford Parto                     | M      | Senior Engineer |
|  10004 | Koblick Chirstian                 | M      | Engineer        |
|  10004 | Koblick Chirstian                 | M      | Senior Engineer |
+--------+-----------------------------------+--------+-----------------+

mysql> select employees.emp_no,
    -> concat(last_name,' ', first_name) as emp_name, gender, title  -- 对名字的列取一个别名叫emp_name
    -> from employees,titles
    -> where employees.emp_no = titles.emp_no limit 5;
+--------+-------------------+--------+-----------------+  -- 这里就显示了emp_name
| emp_no | emp_name          | gender | title           |
+--------+-------------------+--------+-----------------+
|  10001 | Facello Georgi    | M      | Senior Engineer |
|  10002 | Simmel Bezalel    | F      | Staff           |
|  10003 | Bamford Parto     | M      | Senior Engineer |
|  10004 | Koblick Chirstian | M      | Engineer        |
|  10004 | Koblick Chirstian | M      | Senior Engineer |
+--------+-------------------+--------+-----------------+
5 rows in set (0.00 sec)

mysql> select e.emp_no,  -- 使用表的别名
    -> concat(last_name,' ', first_name) as emp_name, gender, title
    -> from employees as e,titles as t      -- 对表做别名
    -> where e.emp_no = t.emp_no limit 5;   -- 使用报表的别名
+--------+-------------------+--------+-----------------+
| emp_no | emp_name          | gender | title           |
+--------+-------------------+--------+-----------------+
|  10001 | Facello Georgi    | M      | Senior Engineer |
|  10002 | Simmel Bezalel    | F      | Staff           |
|  10003 | Bamford Parto     | M      | Senior Engineer |
|  10004 | Koblick Chirstian | M      | Engineer        |
|  10004 | Koblick Chirstian | M      | Senior Engineer |
+--------+-------------------+--------+-----------------+
5 rows in set (0.00 sec)

--
-- ANSI SQL 92
-- inner join ... on ...语法
--
mysql> select e.emp_no,
    -> concat(last_name,' ', first_name) as emp_name, gender, title
    -> from employees as e inner join titles as t  -- inner join 可以省略inner关键字
    -> on e.emp_no = t.emp_no limit 5;             -- 配合join使用on
+--------+-------------------+--------+-----------------+
| emp_no | emp_name          | gender | title           |
+--------+-------------------+--------+-----------------+
|  10001 | Facello Georgi    | M      | Senior Engineer |
|  10002 | Simmel Bezalel    | F      | Staff           |
|  10003 | Bamford Parto     | M      | Senior Engineer |
|  10004 | Koblick Chirstian | M      | Engineer        |
|  10004 | Koblick Chirstian | M      | Senior Engineer |
+--------+-------------------+--------+-----------------+
5 rows in set (0.00 sec)

--
-- 上面两种语句在效率上其实是一样的，只是语法上的区别
--
--- 第一种
mysql> explain select e.emp_no,
    -> concat(last_name,' ', first_name) as emp_name, gender, title
    -> from employees as e,titles as t
    -> where e.emp_no = t.emp_no limit 5;
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref               | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+-------------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL              | 298124 |   100.00 | NULL        |
|  1 | SIMPLE      | t     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      1 |   100.00 | Using index |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)

--- 第二种
mysql> explain select e.emp_no,
    -> concat(last_name,' ', first_name) as emp_name, gender, title
    -> from employees as e inner join titles as t
    -> on e.emp_no = t.emp_no limit 5;
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref               | rows   | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+-------------+
|  1 | SIMPLE      | e     | NULL       | ALL  | PRIMARY       | NULL    | NULL    | NULL              | 298124 |   100.00 | NULL        |
|  1 | SIMPLE      | t     | NULL       | ref  | PRIMARY       | PRIMARY | 4       | employees.e.emp_no |      1 |   100.00 | Using index |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------+--------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)

-- 通过explain看两条语句的执行计划，发现是一样的，所以性能上是一样的，只是语法的不同
```

4.2. OUTER JOIN

```
--
-- 左连接 left join
--
mysql> use burn_test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table test_left_join_1(a int);
Query OK, 0 rows affected (0.16 sec)

mysql> create table test_left_join_2(b int);
Query OK, 0 rows affected (0.14 sec)

mysql> insert into test_left_join_1 values (1);
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_left_join_1 values (2);
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_left_join_2 values (1);
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_left_join_1;
+------+
| a    |
+------+
|    1 |
|    2 |
+------+
2 rows in set (0.00 sec)

mysql> select * from test_left_join_2;
+------+
| b    |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql> select * from
    -> test_left_join_1 as t1
    -> left join   -- 使用left join
    -> test_left_join_2 as t2
    -> on t1.a = t2.b;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |  -- 满足条件的，显示t2中该条记录的值
|    2 | NULL |  -- 不满足条件的，用NULL填充
+------+------+
2 rows in set (0.00 sec)
-- left join ：  左表 left join 右表 on 条件：
--               左表全部显示，右表是匹配表，
--               如果右表的某条记录满足 [on 条件] 匹配，则该记录显示
--               如果右表的某条记录 不 满足 匹配，则该记录显示NULL

--
-- 右连接 right join （继续使用test_left_join_1和2两张表）
--
mysql> select * from
    -> test_left_join_1 as t1
    -> right join   -- 使用right join
    -> test_left_join_2 as t2
    -> on t1.a = t2.b;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |   -- 右表（t2）全部显示
+------+------+
1 row in set (0.00 sec)
-- right join ：  左表 right join 右表 on 条件
--                右表全部显示，左边是匹配表
--                同样和left join，满足则显示，不满足且右表中有值，则填充NULL

mysql> insert into test_left_join_2 values (3);  -- t2 中再增加一条记录
Query OK, 1 row affected (0.03 sec)

mysql> select * from
    -> test_left_join_1 as t1
    -> right join
    -> test_left_join_2 as t2
    -> on t1.a = t2.b;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |
| NULL |    3 | -- 右表存在，左表没有，用NULL填充
+------+------+
2 rows in set (0.00 sec)

--
-- 查找在t1表，而不在t2表的数据
--
mysql> select * from
    -> test_left_join_1 as t1
    -> left join
    -> test_left_join_2 as t2
    -> on t1.a = t2.b where t2.b is null;
+------+------+
| a    | b    |
+------+------+
|    2 | NULL |  -- 数据1 在t1和t2中都有，所以不显示
+------+------+
1 row in set (0.00 sec)


-- left join ：  left outer join , outer关键字可以省略
-- right join:  right outer join , outer 关键字可以省略

-- join无论inner还是outer，列名不需要一样，甚至列的类型也可以不一样，会进行转换。
-- 一般情况下，表设计合理，需要关联的字段类型应该是一样的


--
-- 查找哪些员工不是经理
--
mysql> select e.emp_no,
    -> concat(last_name,' ', first_name) as emp_name, gender, d.dept_no
    -> from employees as e left join dept_manager as d
    -> on e.emp_no = d.emp_no
    -> where d.emp_no is null limit 5;
+--------+-------------------+--------+---------+
| emp_no | emp_name          | gender | dept_no | -- dept_no是dept_manager的字段
+--------+-------------------+--------+---------+
|  10001 | Facello Georgi    | M      | NULL    |
|  10002 | Simmel Bezalel    | F      | NULL    |
|  10003 | Bamford Parto     | M      | NULL    |
|  10004 | Koblick Chirstian | M      | NULL    |
|  10005 | Maliniak Kyoichi  | M      | NULL    |
+--------+-------------------+--------+---------+
5 rows in set (0.00 sec)

-- 在 inner join中，过滤条件放在where或者on中都是可以的
-- 在 outer join中  条件放在where和on中是不一样的
mysql> select * from
    -> test_left_join_1 as t1
    -> left join
    -> test_left_join_2 as t2
    -> on t1.a = t2.b
    -> where t2.b is null;
+------+------+
| a    | b    |
+------+------+
|    2 | NULL |
+------+------+
1 row in set (0.00 sec)

mysql> select * from
    -> test_left_join_1 as t1
    -> left join
    -> test_left_join_2 as t2
    -> on t1.a = t2.b
    -> and t2.b is null;  -- 除了a=b，还要找到b=null的，但是b里面没有null，所有a全部显示，b全为null
+------+------+
| a    | b    |
+------+------+
|    1 | NULL |
|    2 | NULL |
+------+------+
2 rows in set (0.00 sec)

-- ON 参与outer join的结果的生成，而where只是对结果的一个过滤

--
-- 作业：查处普通员工的title，部门名称，薪资
--
```

4.3. GROUP BY

```
--
-- 找出同一个部门的员工数量
--
mysql> select dept_no, count(dept_no)  -- count是得到数量，这里就是分组函数
    -> from dept_emp
    -> group by dept_no;  -- 通过 dept_no 分组
+---------+----------------+
| dept_no | count(dept_no) |
+---------+----------------+
| d001    |          20211 |
| d002    |          17346 |
| d003    |          17786 |
| d004    |          73485 |
| d005    |          85707 |
| d006    |          20117 |
| d007    |          52245 |
| d008    |          21126 |
| d009    |          23580 |
+---------+----------------+
9 rows in set (0.10 sec)

--
-- 选出部门人数 > 50000
--
mysql> select dept_no, count(dept_no)
    -> from dept_emp
    -> group by dept_no
    -> having count(dept_no) > 50000;  -- 如果是对分组的聚合函数做过滤，使用having，用where报语法错误
+---------+----------------+
| dept_no | count(dept_no) |
+---------+----------------+
| d004    |          73485 |
| d005    |          85707 |
| d007    |          52245 |
+---------+----------------+
3 rows in set (0.09 sec)

--
-- 每个用户每个月产生的订单数目
--
mysql> desc orders;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| o_orderkey      | int(11)     | NO   | PRI | NULL    |       |  -- 订单ID
| o_custkey       | int(11)     | YES  | MUL | NULL    |       |  -- 客户ID
| o_orderstatus   | char(1)     | YES  |     | NULL    |       |
| o_totalprice    | double      | YES  |     | NULL    |       |
| o_orderDATE     | date        | YES  | MUL | NULL    |       |  -- 订单日期
| o_orderpriority | char(15)    | YES  |     | NULL    |       |
| o_clerk         | char(15)    | YES  |     | NULL    |       |
| o_shippriority  | int(11)     | YES  |     | NULL    |       |
| o_comment       | varchar(79) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)

mysql> select  o_orderkey, o_custkey, o_orderDATE from orders limit 3;
+-----------+-----------+-------------+
| o_orderkey | o_custkey | o_orderDATE |
+-----------+-----------+-------------+
|          1 |     36901 | 1996-01-02  |
|          2 |     78002 | 1996-12-01  |
|          3 |    123314 | 1993-10-14  |
+-----------+-----------+-------------+
3 rows in set (0.00 sec)


--
-- 查找客户每年每月产生的订单数
--

mysql> select o_custkey, count(o_orderkey),                                    -> year(o_orderDATE), month(o_orderDATE)
    -> from orders
    -> group by o_custkey, year(o_orderDATE), month(o_orderDATE)
    -> limit 10;
+-----------+------------------+------------------+--------------------+
| o_custkey | count(o_orderkey) | year(o_orderDATE) | month(o_orderDATE) |
+-----------+------------------+------------------+--------------------+
|         1 |                1 |             1992 |                  4 |
|         1 |                1 |             1992 |                  8 |
|         1 |                1 |             1996 |                  6 |
|         1 |                1 |             1996 |                  7 |
|         1 |                1 |             1996 |                 12 |
|         1 |                1 |             1997 |                  3 |
|         2 |                1 |             1992 |                  4 |
|         2 |                1 |             1994 |                  5 |
|         2 |                1 |             1994 |                  8 |
|         2 |                1 |             1994 |                 12 |
+-----------+------------------+------------------+--------------------+
10 rows in set (8.97 sec)

-- 使用 date_format 函数

mysql> select o_custkey, count(o_orderkey),
    -> date_format(o_orderDATE, '%Y-%m') as date
    -> from orders
    -> group by o_custkey, date_format(o_orderDATE, '%Y-%m')
    -> limit 10;
+-----------+------------------+---------+
| o_custkey | count(o_orderkey) | date    |
+-----------+------------------+---------+
|         1 |                1 | 1992-04 |
|         1 |                1 | 1992-08 |
|         1 |                1 | 1996-06 |
|         1 |                1 | 1996-07 |
|         1 |                1 | 1996-12 |
|         1 |                1 | 1997-03 |
|         2 |                1 | 1992-04 |
|         2 |                1 | 1994-05 |
|         2 |                1 | 1994-08 |
|         2 |                1 | 1994-12 |
+-----------+------------------+---------+
10 rows in set (11.46 sec)


-- 作业：查找客户每周（以年，月，周 显示）产生的订单量
```