SQL

1. 创建数据库

```
CREATE DATABASE 数据库名;
```

2. 创建数据表

```
CREATE TABLE table_name (列名 列类型);
```

3. 删除数据表

```
DROP TABLE table_name ;
```

4. 增，（可以同时插入多条）

```
INSERT INTO table_name ( field1, field2,...fieldN )
 VALUES
 ( value1, value2,...valueN ),
 ( value1, value2,...valueN ),
```

5. 删

```
DELETE FROM table_name [WHERE Clause]
```

如果没有*WHERE...*就删除全部。关于删除表数据，drop、truancate和delete的区别：

```
drop table table_name，删除内容和定义，释放空间；
truncate table table_name，删除内容，释放空间但不删除定义(保留了表结构)；
delete，删除一行，并且同时将改行的删除操作作为事务记录在日志保存，可以进行回滚。
```

6. 改

```
UPDATE table_name SET
field1=new-value1, field2=new-value2
[WHERE Clause]
```

7. 查***

```
SELECT column_name,column_name
FROM table_name
[WHERE Clause]
```

```
[LIMIT N][ OFFSET M][ORDER BY column_name]
```

LIMIT 子句用于限制查询结果返回的数量limit i,n 表示从第i个开始，第n个结束
OFFSET 表示跳过数据，offset m 表示跳过m条数据 ,在navicat上没测试成功...
limit i,n 等同于 limit n offset i

8. LIKE 子句

```
SELECT field1, field2,...fieldN
FROM table_name
WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

LIKE可以用来匹配文本：

like "%a"，代表以a为结尾的数据
like "a%"，代表以a为开始的数据
like "%a%"，代表含有a的数据
like "_a_"，三位且中间字母是a的数据
like "_a"，两位且结尾字母是a的数据
like "a_"，两位且开始字母是a的数据

9. UNION

UNION 操作符用于连接两个以上的 SELECT 语句的结果组合到一个结果集合中。多个 SELECT 语句会删除重复的数据。

```
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions]
UNION [ALL | DISTINCT]
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions];
```

ALL代表全部，包含重复的；DISTINCT只包含不重复；

10. 排序

```
SELECT field1, field2,...fieldN FROM table_name1, table_name2...
ORDER BY field1 [ASC [DESC][默认 ASC]], [field2...] [ASC [DESC][默认 ASC]]
```

ASC，默认，代表升序；DESC，代表逆序；

11. 分组

GROUP BY 语句根据一个或多个列对结果集进行分组。
在分组的列上我们可以使用 COUNT, SUM, AVG,等函数。

```
SELECT column_name, function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
```

WITH ROLLUP 可以实现在分组统计数据基础上再进行相同的统计（SUM,AVG,COUNT…）。就是在统计结果上面再加一行

```
SELECT name, SUM(singin) as singin_count
FROM employee_tbl
GROUP BY name
WITH ROLLUP;
```

with rollup统计的最后一行是没有名字的，希望自定义名字可以使用coalesce(a, b, c);表示当a==null的时候，选择b，当a==b==null的时候，选择c，当全部都为null的时候，返回null。上面的代码可以优化成：

```
SELECT coalesce(name, '总数'), SUM(singin) as singin_count
FROM employee_tbl
GROUP BY name
WITH ROLLUP;
```

12. 连接 JOIN

INNER JOIN (内连接)，获取两个表中字段匹配关系的记录
LEFT JOIN（左连接），获取左表所有记录，即使右表没有对应的记录
RIGHT JOIN（右连接），获取右表所有记录，及时左表没有对应的记录

例子：

```
SELECT a.runoob_id, a.runoob_author, b.runoob_count
FROM runoob_tbl a INNER JOIN tcount_tbl b
ON a.runoob_author = b.runoob_author;
```

等价于：

```
SELECT a.runoob_id, a.runoob_author, b.runoob_count
```

```
FROM runoob_tbl a, tcount_tbl b
WHERE a.runoob_author = b.runoob_author;
```

左连接

```
SELECT a.runoob_id, a.runoob_author, b.runoob_count
FROM runoob_tbl a LEFT JOIN tcount_tbl b
ON a.runoob_author = b.runoob_author;
```

右连接

```
SELECT a.runoob_id, a.runoob_author, b.runoob_count
FROM runoob_tbl a RIGHT JOIN tcount_tbl b
ON a.runoob_author = b.runoob_author;
```

13. NULL值的处理

> 空值不能用“==”，要用IS NULL,IS NOT NULL, 或者<=>

14. 事务

MySQL的事务处理机制，就是批处理。事务包含的操作，要么不做，要么全做。描述一个场景：
在爬虫爬取一个页面的时候，如果爬取到一半出现错误，数据库只保存了前一半的内容，而页面需要重新爬取，那么就不能保存前半部分的内容，保存了的最好也要删除。因此，可以利用事务的回滚ROLLBACK，将前半部分保存的也取消。

> 1. BEGIN 或 START TRANSACTION 显式地开启一个事务；
> 2. COMMIT 也可以使用 COMMIT WORK，不过二者是等价的。COMMIT 会提交事务，并使已对数据库进行的所有修改成为永久性的；
> 3. ROLLBACK 也可以使用 ROLLBACK WORK，不过二者是等价的。回滚会结束用户的事务，并撤销正在进行的所有未提交的修改；
> 4. SAVEPOINT identifier，SAVEPOINT 允许在事务中创建一个保存点，一个事务中可以有多个 SAVEPOINT；
> 5. RELEASE SAVEPOINT identifier 删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常；
> 6. ROLLBACK TO identifier 把事务回滚到标记点；
> 7. SET TRANSACTION 用来设置事务的隔离级别。InnoDB 存储引擎提供事务的隔离级别有READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ 和 SERIALIZABLE。

15. ALTER

> 修改数据表的名称或者数据表的字段

```
# 删除字段
ALTER TABLE table_name DROP column_name;
# 增加字段
ALTER TABLE table_name ADD column_name column_type;
# 将新增字段变为第一列
ALTER TABLE table_name ADD column_name column_type FIRST;
# 将新增字段变为任意一列之后
ALTER TABLE table_name ADD column_name column_type AFTER other_colmun_name;
# 修改字段类型
ALTER TABLE table_name MODIFY column_name column_type;
ALTER TABLE table_name CHANGE old_column_name new_column_name column_type;
# 将字段设置成非空默认值
ALTER TABLE table_name MODIFY column_name column_type NOT NULL DEFAULT 1;
# 修改字段默认值
ALTER TABLE table_name ALTER column_name SET DEFAULT 2;
# 删除字段默认值
ALTER TABLE table_name ALTER column_name DROP DEFAULT;
# 修改表名
ALTER TABLE table_name RENAME TO other_name;
```

16. 临时表、复制表

> 临时表只在当前数据库连接中存在，连接关闭会删除并释放空间。

```
# 创建临时表
CREATE TEMPORARY table_name (column_name column_type ...)
# 仅复制结构
CREATE TEMPORARY table_name LIKE old_table_name
# 复制结构和数据
CREATE TEMPORARY table_name AS (SELECT ... FROM table_name WHERE ...)
```

复制表操作相同，去掉TEMPORARY即可。

17. 自增序列

> MySQL 序列是一组整数：1, 2, 3, ...，由于一张数据表只能有一个字段自增主键， 如果你想实现其他字段也实现自动增加，就可以使用MySQL序列来实现。

```
# 创建一个带自增字段的表
CREATE TABLE insect
 (
 id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 PRIMARY KEY (id),
 name VARCHAR(30) NOT NULL
 );
# 重置序列，先删除，然后重新添加
 ALTER TABLE insect DROP id;
ALTER TABLE insect
ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT FIRST,
ADD PRIMARY KEY (id);
# 设置初始值
ALTER TABLE table_name AUTO_INCREMENT = 100;
```

18. 重复数据处理

```
# 查找重复数据
SELECT COUNT(*) AS a, column_name1, column_name2 FROM table_name
GROUP BY column_name1, column_name2
HAVING a > 1;

# 过滤重复数据
SELECT DISTINCT column_name1,column_name2 FROM table_name

SELECT column_name1, column_name2 FROM table_name
 GROUP BY (column_name1, column_name2);

# 删除重复数据，通过添加索引和主键
ALTER IGNORE TABLE table_name ADD PRIMARY KEY (column_name1, column_name2);
```

19. 字符串函数

```
ASCII(s) # 返回字符串s的第一个字符的ASCII码
SELECT ASCII(CustomerName) AS NumCode OfFirstChar FROM Customers;

CHAR_LENGTH(s) # 返回字符串s的字符数;
CHARACTER_LENGTH(s) # 效果等同于上个;
# 这两个与LENGTH的区别，在于：length统计的是字节数。
# s是单字节，结果相同；s如果不是单字节，例如中文，则length的结果是CHAR_LENGTH的三倍。

CONCAT(s1,s2...sn) # 将字符串s1,s2...sn合并为一个字符串;
 CONCAT_WS(x, s1, s2...sn) # 将字符串合并，并在每个字符串之间添加分隔符x;

 FILED(s,s1,s2...sn) # 返回字符s在字符串列表(s1,s2...sn)中的位置;
 SELECT FIELD("c", "a", "b", "c", "d", "e");
 # 输出：3

 FIND_IN_SET(s1, s2) # 返回在字符串s2中s1匹配的字符串的位置;
 SELECT FIND_IN_SET("c", "a,b,c,d,e");
 # 输出：3

 FORMAT(x, n) # 将数字x格式化为小数点后n位，最后一位四舍五入，不够则补0;

 INSERT(s1, x, len, s2) # 字符串s2替换s1的x位置开始，长度为len的字符串
 SELECT INSERT("google.com", 1, 6, "runnob");
 # 输出：runoob.com

 LCASE(s) # 将字符串s转换为小写
 LOWER(s) # 将字符串s转换为小写
 UCASE(s) # 将字符串s转换为大写
 UPPER(s) # 将字符串s转换为大写

 LEFT(s, n) # 返回字符串s的前n个字符
 RIGHT(s, n) # 与LEFT相反
 SELECT LEFT('runoob',2)
 # 输出：  ru
```

```
35
36 LPAD(s1, len, s2) # 在字符串s1的开始处填充字符串s2，使得字符串长度达到len
37 RPAD(s1, len, s2) # 效果同上，方向在右
38 SELECT LPAD('abc', 5, 'xx')
39 # 输出： xxabc
40
41 LTRIM(s) # 去掉字符串s开始处的空格
42 RTRIM(s) # 效果同上，方向在右
43 TRIM(s) # 效果同上，方向在右
44
45 MID(s, n, len) # 从字符串s的n位置截取长度为len的子字符串
46 SUBSTR(s, n, length) # 与上述函数相同
47 SUBSTRING(s, n, length) # 与上述函数相同
48 SELECT MID("RUNOOB", 2, 3);
49 # 输出: UNO
50
51 POSITION(s1 IN s) # s1在s中开始的位置，如果不存在，就返回0
52 SELECT POSITION('b' in 'abc');
53 # 输出: 2
54 SELECT POSITION('d' in 'abc');
55 # 输出: 0
56
57 REPLACE(s, s1, s2) # 将字符串s中的s1替换成s2，相当于python中的replace
58
59 REVERSE(s) # 将字符串s翻转
60
61 SPACE(n) # 返回n个空格
62
63 SUBSTRING_INDEX(s, delimiter, number) # delimiter是分隔符，number代表索引
64 # 该函数表示，按照delimiter切分是s， 获取第number个字符串，number可以为负
65 SELECT SUBSTRING_INDEX('a*b','*',1)
66 # 输出:a
67 SELECT SUBSTRING_INDEX('a*b','*',-1)
68 # 输出: b
69 SELECT SUBSTRING_INDEX(SUBSTRING_INDEX('a*b*c*d*e','*',3),'*',-1)
70 # 输出: c
```

20. 数字函数

```
# 列操作
AVG(column_name) # 求平均值
COUNT(column_name) # 计数
MAX(column_name) # 最大值
MIN(column_name) # 最小值
SUM(column_name) # 求和

# 数学计算
ABS(x) # 绝对值
CEIL(x), CEILING(x) # 向上取整
FLOOR(x) # 向下取整
ROUND(x) # 四舍五入取整
TRUNCATE(x, y) # 返回数值x，保留到小数点后y位，直接舍弃y位后面的，不够就补0。
SELECT CEIL(1.5)
# 输出：2
 SELECT TRUNCATE(1.2345, 3);
 # 输出：1.234
n DIV m # 整除
SELECT 10 DIV 5;
# 输出：2
EXP(x) # e的x次方
LOG(x) # x的自然对数； 其他对数：LOG10(x)，LOG2(x)
SELECT LOG(EXP(5));
# 输出：5
MOD(x,y) # 取余
SELECT MOD(5,2);
# 输出：1
POW(x, y) POWER(x, y) # 返回x的y次方
RAND() # 返回0到1之间的随机数
SIGN(x) # 返回x的正负
SELECT SIGN(5);
# 输出：1
SELECT SIGN(-5);
# 输出：-1
SQRT(x) # 返回x的平方根

```

```
# 三角函数计算
COS(x) # 求x的余弦值，x是弧度
ACOS(x) # 求x的反余弦值，x是余弦值，超过0-1的区间，返回NULL
SELECT cos(0);
# 输出：1
SELECT ACOS(2);
# 输出 NULL
# 其他类似
SIN(x)，ASIN(x) # 正弦
TAN(x), ATAN(x) # 正切
COT(x) # 余切
DEGREES(x) # 将弧度转换为角度
RADIANS(x) # 将角度转换为弧度
SELECT DEGREES(3.1415926);
# 输出：179.99999692953102
```

21. 日期函数

```
# 获取当前日期、时间
CURDATE() CURRENT_DATE() # 返回当前日期
CURTIME() CURRENT_TIME() # 返回当前时间
CURRENT_TIMESTAMP() # 返回当前时间和日期
NOW() # 返回当前日期和时间
LOCALTIMESTAMP() LOCALTIME() LOCALTIMESTAMP() # 返回当前日期和时间
SYSDATE()


# 日期、时间的计算
ADDDATE(d, INTERVAL expr type) # 计算起始时期d加上一个时间段后的日期
DATE_ADD(d, INTERVAL expr type) # 效果用法同上
SUBDATE(d, n) # 效果与上述相反，减法
ADDTIME(t, n) # 时间t加上n秒
SUBTIME(t,n) # 与上述相反，减法
select adddate('2018-06-15', 10);
# 输出：2018-06-25
select adddate('2018-06-15 17:56:20', INTERVAL 5 MINUTE);
# 输出：2018-06-15 18:01:20
DATEDIFF(d1, d2) # 计算d1->d2之间相隔的天数
```

```
PERIOD_DIFF(period_1, period_2) # 返回两个时段之间的月份差值
TIMEDIFF(t1, t2) # 时间差值
SELECT DATEDIFF('2001-01-01','2001-02-02')
# 输出：32
PERIOD_ADD(period, number) # 为年-月 组合日期添加一个时段
SELECT PERIOD(201901,  6)
# 输出：201907
SEC_TO_TIME(s) # 将以秒为单位的时间s转换为时分秒的格式
TIME_TO_SEC(t) # 上面操作的反向操作

# 日期、时间提取
DATE(x) # 从日期或日期时间表达式中提取日期值
SELECT DATE("2019-12-19")
# 输出：2019-12-19
DATE_FORMAT(d, f) # 按表达式f显示日期d
SELECT DATE_FORMAT('2011-11-11 18:11:11','%Y-%m-%d %r');
# 输出：2011-11-11 06:11:11 PM
DAY(d) # 返回日期值d的天数
HOUR(t) # 获取t中小时值
MINUTE(t) # 获取t中的分钟数
SECOND(t) # 获取t中秒数
MONTH(d) # 获取d中的月份
WEEK(d) # 当年周数
QUARTER(d) # 返回日期d的第几季节
DAYNAME(d) # 返回周几，英文
MONTHNAME(d) # 返回月份的名称，英文
DAYOFMONTH(d) DAYOFYEAR(d) DAYOFWEEK(d) # 返回d在这个月、这一年、这周的第几天
SELECT DAYOFYEAR('2019-11-11 11:11:11')
# 输出315
EXTRACT(type FROM d) # 从日期d中获取指定的值，type指定返回的值
SELECT EXTRACT(MINUTE FROM '2011-11-11 11:11:11')
# type值为：MICROSECOND,SECOND,MINUTE,HOUR,DAY,WEEK,MONTH,QUARTER,YEAR;

# 自定义设置日期、日期
MAKEDATE(year, day-of-year) # 基于给定参数年份year和
 # 所在年中的天数序号day-of-year返回一个日期
SELECT MAKEDATE(2018, 40);
```

```
# 输出：2018-02-09
MAKETIME(hour, minute, second) # 组合时间，小时、分钟、秒
```

22. 其他函数

条件语句：CASE函数

CASE表示函数开始，END表示结束，相当于JAVA的switch case语句，当满足其中一个条件，后面的就不再执行。

```
CASE expression
 WHEN condition1
 THEN result1
 WHEN condition2
 THEN result2
 ...
 ELSE result
END
```

NULLIF

比较两个字符串，如果字符串相等，返回NULL，否则返回expr1

```
NULLIF(expr1， expr2)
SELECT NULLIF('abc', 'abc');
# 输出： NULL
SELECT NULLIF('abc', 'ab');
# 输出：abc
```

23. 一些简单组合问题

- 数据库包含a1, a2两列，求两列之和排名第二的数据

```
SELECT (a1+a2) as A FROM test ORDER BY A DESC LIMIT 1, 1;
```