一、数据库与数据表的基本操作

1. 创建与查看数据库

```
-- 创建数据库：指定数据库名称
CREATE DATABASE 数据库名;

-- 查看当前所有数据库
SHOW DATABASES;

-- 切换到指定数据库
USE 数据库名;
```

2. 删除数据库

```
-- 删除数据库（如果存在才删除）
DROP DATABASE IF EXISTS 数据库名;
```

3. 创建数据表

```
CREATE TABLE 表名 (
    列名1 数据类型 [列级约束]  COMMENT '注释',
    列名2 数据类型 [列级约束]  COMMENT '注释',
    …,
    [表级约束]
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='表的说明';
```

- **示例**：

```
CREATE TABLE users (
    user_id    INT NOT NULL AUTO_INCREMENT COMMENT '用户唯一标识',
    username   VARCHAR(50) NOT NULL UNIQUE      COMMENT '用户名',
    password   VARCHAR(255) NOT NULL            COMMENT '哈希密码',
    email      VARCHAR(100)                     COMMENT '邮箱地址',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    PRIMARY KEY (user_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户表';
```

二、常用数据类型

数据类型	描述

数据类型	描述
`INT(size)`、`SMALLINT`、`TINYINT`	整数类型，`size` 指显示宽度（不影响存储范围）
`DECIMAL(size, d)`、`NUMERIC(size,d)`	精确小数，`size` 为总位数，`d` 为小数位数
`CHAR(n)`	定长字符串，长度固定为 `n`
`VARCHAR(n)`	变长字符串，最大长度为 `n`
`TEXT`、`MEDIUMTEXT`、`LONGTEXT`	大型文本类型，根据容量依次增大
`DATE`	'YYYY-MM-DD' 格式的日期
`DATETIME`、`TIMESTAMP`	日期时间类型：`DATETIME` 范围更大，`TIMESTAMP` 可自动更新
`BOOLEAN`	布尔类型，MySQL 中为 `TINYINT(1)` 的别名

三、列级约束与表级约束

1. 列级约束

- **NOT NULL**
  列值不允许为 NULL。

```
ALTER TABLE users ADD age INT NOT NULL;
```

- **DEFAULT**
  列的默认值。

```
ALTER TABLE users ADD is_active BOOLEAN DEFAULT TRUE;
```

- **UNIQUE**
  列内值必须唯一。

```
ALTER TABLE users ADD email VARCHAR(100) UNIQUE;
```

- **CHECK**
  列值必须满足条件。

```
ALTER TABLE users ADD age INT CHECK (age > 0);
```

- **AUTO_INCREMENT**
  自增列，通常用于主键。

```
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT,
    …
    PRIMARY KEY (order_id)
);
```

2. 表级约束

- **PRIMARY KEY**
  唯一标识表中行，自动隐含 `NOT NULL` 与 `UNIQUE`。

```
CREATE TABLE student_course (
    Sno   VARCHAR(20),
    Cno   VARCHAR(20),
    Grade DECIMAL(5,2),
    PRIMARY KEY (Sno, Cno)        -- 复合主键
);
```

- **FOREIGN KEY**
  外键约束，确保引用完整性。

```
ALTER TABLE orders
ADD CONSTRAINT fk_orders_user
FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE;
```

四、修改与删除表结构

1. 添加列

```
ALTER TABLE table_name ADD 列名 数据类型 [列级约束];
```

2. 修改列

- **修改数据类型**

```
ALTER TABLE table_name MODIFY COLUMN 列名 新数据类型 [列级约束];   -- MySQL
ALTER TABLE table_name ALTER COLUMN 列名 TYPE 新数据类型;         --
PostgreSQL
```

- **重命名列**

```
ALTER TABLE table_name CHANGE old_name new_name 数据类型;          -- MySQL
ALTER TABLE table_name RENAME COLUMN old_name TO new_name;        --
PostgreSQL
```

3. 删除列

```
ALTER TABLE table_name DROP COLUMN 列名;
```

4. 添加/删除主键

```
ALTER TABLE table_name ADD PRIMARY KEY (列名);
ALTER TABLE table_name DROP PRIMARY KEY;
```

五、索引与视图

1. 索引

- **创建索引**

```
CREATE INDEX idx_username ON users(username);
```

- **删除索引**

```
DROP INDEX idx_username ON users;
```

> **Tip：** 合理使用索引能大幅提升查询性能，但过多索引会拖慢写入速度。

2. 视图

- **创建视图**

```
CREATE VIEW active_users AS
SELECT user_id, username, email
FROM users
WHERE is_active = TRUE;
```

- **删除视图**

```
DROP VIEW IF EXISTS active_users;
```

六、数据操作语言（DML）

1. 插入数据

- **指定列插入**

```
INSERT INTO table_name (col1, col2, …)
VALUES (val1, val2, …);
```

- **不指定列**

```
INSERT INTO table_name VALUES (val1, val2, …);
```

- **批量插入多行**

```
INSERT INTO table_name (col1, col2)
VALUES (v1a, v1b),
       (v2a, v2b),
       …;
```

- **从其它表插入**

```
INSERT INTO table2 (col1, col2)
SELECT colA, colB FROM table1 WHERE …;
```

2. 更新数据

```
UPDATE table_name
SET col1 = val1,
    col2 = val2,
    …
WHERE 条件;
```

3. 删除数据

- **删除部分行**

```
DELETE FROM table_name WHERE 条件;
```

- **清空整表**

```
TRUNCATE TABLE table_name;
```

七、查询数据

1. 基本查询

```
SELECT * FROM table_name;
SELECT col1, col2 FROM table_name;
```

2. 条件与排序

- **WHERE**

```
SELECT * FROM employees WHERE salary > 5000;
```

- **LIKE**

```
SELECT * FROM users WHERE username LIKE 'J%';
```

- **BETWEEN / IN**

```
SELECT * FROM orders WHERE created_at BETWEEN '2025-01-01' AND '2025-03-31';
SELECT * FROM employees WHERE department_id IN (10,20,30);
```

- **ORDER BY**

```
SELECT * FROM products ORDER BY price DESC, name ASC;
```

3. 分组与聚合

```
SELECT department_id, COUNT(*) AS cnt, AVG(salary) AS avg_sal
FROM employees
```

```
GROUP BY department_id
HAVING COUNT(*) > 5;
```

- 常用聚合函数：`SUM()`, `AVG()`, `COUNT()`, `MAX()`, `MIN()`

4. 连接查询

```
-- 内连接
SELECT e.employee_id, e.name, d.department_name
FROM employees e
INNER JOIN departments d
  ON e.department_id = d.department_id;

-- 左连接
SELECT e.employee_id, e.name, d.department_name
FROM employees e
LEFT JOIN departments d
  ON e.department_id = d.department_id;

-- 全连接（MySQL 可用 UNION 模拟）
SELECT * FROM (
  SELECT e.*, d.department_name FROM employees e LEFT JOIN departments d ON
e.department_id=d.department_id
  UNION
  SELECT e.*, d.department_name FROM employees e RIGHT JOIN departments d ON
e.department_id=d.department_id
) AS t;
```

5. 子查询

```
SELECT * FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

八、事务与权限

1. 事务控制

```
START TRANSACTION;       -- 或 BEGIN
-- DML 操作：INSERT/UPDATE/DELETE
COMMIT;                  -- 提交
ROLLBACK;                -- 回滚
```

2. 用户与权限

```
-- 创建用户
CREATE USER 'username'@'host' IDENTIFIED BY 'password';
-- 授权
GRANT SELECT, INSERT ON dbname.tablename TO 'username'@'host';
-- 撤销权限
REVOKE INSERT ON dbname.tablename FROM 'username'@'host';
-- 删除用户
DROP USER 'username'@'host';
```