存储函数

1. 返回员工总人数

```
DELIMITER //
CREATE FUNCTION GetTotalEmployees()
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO total FROM Employees;
    RETURN total;
END //
DELIMITER ;
```

2. 删除无效员工号（改用存储过程）

```
DELIMITER //
CREATE PROCEDURE DeleteInvalidEmployee(IN emp_id CHAR(6), OUT result BOOLEAN)
BEGIN
    DECLARE exists_flag INT;
    SELECT COUNT(*) INTO exists_flag FROM Employees WHERE EmployeeID = emp_id;
    IF exists_flag > 0 THEN
        SET result = FALSE;
    ELSE
        DELETE FROM Salary WHERE EmployeeID = emp_id;
        SET result = TRUE;
    END IF;
END //
DELIMITER ;
```

3. 判断是否在研发部并返回学历

```
DELIMITER //
CREATE FUNCTION CheckResearchDept(emp_id CHAR(6))
    RETURNS VARCHAR(255)
    DETERMINISTIC
BEGIN
    DECLARE dept_name CHAR(20);
    DECLARE edu CHAR(4);
    SELECT d.DepartmentName, e.Education INTO dept_name, edu
    FROM Employees e
             JOIN Departments d ON e.DepartmentID = d.DepartmentID
    WHERE e.EmployeeID = emp_id;
    IF dept_name = '研发部' THEN
        RETURN edu;
    ELSE
        RETURN 'NO';
    END IF;
END //
```

```
DELIMITER ;
```

4. **增加工作满四年员工的收入（改用存储过程）**

```
DELIMITER //
CREATE PROCEDURE IncreaseSalaryForFourYears()
BEGIN
    -- 通过 Employees.WorkYear 筛选员工，并关联更新 Salary.Income
    UPDATE Salary s
        JOIN Employees e ON s.EmployeeID = e.EmployeeID
    SET s.Income = s.Income + 500
    WHERE e.WorkYear >= 4;
END //
DELIMITER ;
```

触发器

1. **级联删除员工信息**

```
DELIMITER //
CREATE TRIGGER AfterEmployeeDelete
    AFTER DELETE ON Employees
    FOR EACH ROW
BEGIN
    DELETE FROM Salary WHERE EmployeeID = OLD.EmployeeID;
END //
DELIMITER ;
```

2. **同步插入部门到 Departments2**

```
CREATE TABLE Departments2 LIKE Departments;
DELIMITER //
CREATE TRIGGER AfterDepartmentInsert
    AFTER INSERT ON Departments
    FOR EACH ROW
BEGIN
    INSERT INTO Departments2 (DepartmentID, DepartmentName, Note)
    VALUES (NEW.DepartmentID, NEW.DepartmentName, NEW.Note);
END //
DELIMITER ;
```

3. **根据工作时间调整收入**

```
DELIMITER //
CREATE TRIGGER AfterDepartmentUpdate
    AFTER UPDATE ON Departments
    FOR EACH ROW
BEGIN
    UPDATE Employees
    SET DepartmentID = NEW.DepartmentID
    WHERE DepartmentID = OLD.DepartmentID;
END //
DELIMITER ;
```

4. **同步更新部门信息**

```
DELIMITER //
CREATE TRIGGER AfterDepartmentUpdate
AFTER UPDATE ON Departments
FOR EACH ROW
BEGIN
    UPDATE Employees
    SET DepartmentID = NEW.DepartmentID
    WHERE DepartmentID = OLD.DepartmentID;
END //
DELIMITER ;
```

事件

1. **创建表 `eventlog`**

```
CREATE TABLE eventlog (
                              log_id INT AUTO_INCREMENT PRIMARY KEY,
                              event_type INT,
                              log_time DATETIME
```

2. **立即执行的事件**

```
CREATE EVENT EventImmediate
    ON SCHEDULE AT CURRENT_TIMESTAMP
    DO
    INSERT INTO eventlog (event_type, log_time) VALUES (1, NOW());
```

3. **每两分钟执行的事件**

```
CREATE EVENT EventEveryTwoMinutes
    ON SCHEDULE EVERY 2 MINUTE
        STARTS NOW()
        ENDS '2025-12-31 23:59:59'
    DO
    INSERT INTO eventlog (event_type, log_time) VALUES (2, NOW());
```

4. **特定时间执行的事件**

```
CREATE EVENT EventSpecificTime
    ON SCHEDULE AT '2025-04-03 17:00:00'
    DO
    INSERT INTO eventlog (event_type, log_time) VALUES (3, NOW());
```