프로젝트 6 Hive 데이터 조회(DQL) 종합 실습 워크북

> **과목**: Hive 데이터 웨어하우스 활용

> **실습 시간**: 약 4 시간

> **데이터베이스**: unit6db（경로: /user/hive/warehouse/unit6db）

> **데이터 파일 디렉토리**: /usr/datas/unit6/

> **Beeline 실행 명령어**: `hive-bee`

> **MySQL 정보**: 사용자명 hiveuser, 비밀번호 123456

목차

| 장 | 내용 | 권장 시간 |

|---|---|---|

| 제 1 부 | 환경 준비 및 데이터 로드 | 25 분 |

| 제 2 부 | SELECT 기초 조회（select_expr / ALL / DISTINCT） | 30 분 |

| 제 3 부 | WHERE 절과 파티션 프루닝 |

项目六 Hive 数据查询（DQL）综合实践工作手册

> **课程**：Hive 数据仓库应用

> **实践时间**：约 4 小时

> **数据库**：unit6db（路径：/user/hive/warehouse/unit6db）

> **数据文件目录**：/usr/datas/unit6/

> **Beeline 启动命令**：`hive-bee`

> **MySQL 信息**：用户名 hiveuser，密码 123456

目录

| 章节 | 内容 | 建议时间 |

|---|---|---|

| 第一部分 | 环境准备与数据加载 | 25 分钟 |

| 第二部分 | SELECT 基础查询（select_expr / ALL / DISTINCT） | 30 分钟 |

| 第三部分 | WHERE 子句与分区裁剪 | 25 分钟 |

25 분 |

| 제 4 부 | GROUP BY 와 HAVING | 30 분 |

| 第四部分 | GROUP BY 与 HAVING | 30 分钟 |

| 제 5 부 | LIMIT 와 SQL 실행 순서 | 20 분 |

| 第五部分 | LIMIT 与 SQL 执行顺序 | 20 分钟 |

| 제 6 부 | 고급 정렬 (ORDER BY / CLUSTER BY / DISTRIBUTE BY+SORT BY) | 35 분 |

| 第六部分 | 高阶排序 (ORDER BY / CLUSTER BY / DISTRIBUTE BY+SORT BY) | 35 分钟 |

| 제 7 부 | UNION 연합 조회와 서브쿼리 | 30 분 |

| 第七部分 | UNION 联合查询与子查询 | 30 分钟 |

| 제 8 부 | CTE 공통 테이블 표현식 | 20 분 |

| 第八部分 | CTE 公用表表达式 | 20 分钟 |

| 제 9 부 | JOIN 연결 연산 (6 종) | 45 분 |

| 第九部分 | JOIN 连接操作 (6 种) | 45 分钟 |

| 제 10 부 | 종합 도전 과제 | 40 분 |

| 第十部分 | 综合挑战练习 | 40 分钟 |

제 1 부: 환경 준비 및 데이터 로드 (25 분)

第一部分：环境准备与数据加载（25 分钟）

학습 목표

学习目标

- 실습용 데이터베이스 및 필요한 테이블 구조 생성

- 创建实践数据库及所需表结构

- 데이터 파일을 Hive 테이블에 로드

- 将数据文件加载到 Hive 表中

- 데이터 검증 방법 습득

1.1 데이터 파일 준비

> **작업 설명**: 터미널（Terminal）에서 다음 명령어를 실행하여 데이터 디렉토리를 생성하고 데이터 파일을 만듭니다.

단계 1: 데이터 디렉토리 생성

```
sudo mkdir -p /usr/datas/unit6
sudo chmod 777 /usr/datas/unit6
```

단계 2: 미국 코로나 19 데이터 파일 생성（us_covid19.txt）

```
cat > /usr/datas/unit6/us_covid19.txt << 'EOF'
2021-01-28,Autauga,Alabama,1001,5554,69
2021-01-28,Baldwin,Alabama,1003,17779,225
2021-01-
```

- 掌握数据验证方法

1.1 准备数据文件

> **操作说明**：在终端（Terminal）中执行以下命令，创建数据目录并生成数据文件。

步骤 1：创建数据目录

```
sudo mkdir -p /usr/datas/unit6
sudo chmod 777 /usr/datas/unit6
```

步骤 2：创建美国新冠数据文件（us_covid19.txt）

```
cat > /usr/datas/unit6/us_covid19.txt << 'EOF'
2021-01-28,Autauga,Alabama,1001,5554,69
2021-01-28,Baldwin,Alabama,1003,17779,225
2021-01-
```

```
28,Barbour,Alabama,1005,2083,42
2021-01-
28,Bibb,Alabama,1007,2298,50
2021-01-
28,Blount,Alabama,1009,5849,92
2021-01-28,Los
Angeles,California,6037,1162766,17
096
2021-01-28,San
Diego,California,6073,210605,2340
2021-01-
28,Orange,California,6059,225673,2
538
2021-01-
28,Riverside,California,6065,22229
6,2997
2021-01-28,San
Bernardino,California,6071,258644,
2734
2021-01-28,New York City,New
York,36061,514561,29072
2021-01-28,Nassau,New
York,36059,115703,3011
2021-01-28,Suffolk,New
York,36103,125025,3280
2021-01-28,Erie,New
York,36029,67888,1359
2021-01-28,Westchester,New
York,36119,85698,1614
2021-01-
28,Cook,Illinois,17031,328892,6637
2021-01-
28,DuPage,Illinois,17043,61949,872
2021-01-
28,Lake,Illinois,17097,52248,827
2021-01-
28,Will,Illinois,17197,49891,631
2021-01-
28,Maricopa,Arizona,4013,339843,71
28
2021-01-
28,Pima,Arizona,4019,73748,1471
2021-01-
```


```
28,Barbour,Alabama,1005,2083,42
2021-01-
28,Bibb,Alabama,1007,2298,50
2021-01-
28,Blount,Alabama,1009,5849,92
2021-01-28,Los
Angeles,California,6037,1162766,17
096
2021-01-28,San
Diego,California,6073,210605,2340
2021-01-
28,Orange,California,6059,225673,2
538
2021-01-
28,Riverside,California,6065,22229
6,2997
2021-01-28,San
Bernardino,California,6071,258644,
2734
2021-01-28,New York City,New
York,36061,514561,29072
2021-01-28,Nassau,New
York,36059,115703,3011
2021-01-28,Suffolk,New
York,36103,125025,3280
2021-01-28,Erie,New
York,36029,67888,1359
2021-01-28,Westchester,New
York,36119,85698,1614
2021-01-
28,Cook,Illinois,17031,328892,6637
2021-01-
28,DuPage,Illinois,17043,61949,872
2021-01-
28,Lake,Illinois,17097,52248,827
2021-01-
28,Will,Illinois,17197,49891,631
2021-01-
28,Maricopa,Arizona,4013,339843,71
28
2021-01-
28,Pima,Arizona,4019,73748,1471
2021-01-
```

```
28,Pinal,Arizona,4021,35076,661
2021-01-
28,Harris,Texas,48201,237567,3590
2021-01-
28,Dallas,Texas,48113,186074,2399
2021-01-
28,Tarrant,Texas,48439,149282,1727
2021-01-
28,Bexar,Texas,48029,118094,1689
2021-01-
28,Travis,Texas,48453,72638,725
2021-01-
28,King,Washington,53033,60283,109
1
2021-01-
28,Pierce,Washington,53053,34481,4
26
2021-01-
28,Snohomish,Washington,53061,2903
8,338
EOF
```

```
28,Pinal,Arizona,4021,35076,661
2021-01-
28,Harris,Texas,48201,237567,3590
2021-01-
28,Dallas,Texas,48113,186074,2399
2021-01-
28,Tarrant,Texas,48439,149282,1727
2021-01-
28,Bexar,Texas,48029,118094,1689
2021-01-
28,Travis,Texas,48453,72638,725
2021-01-
28,King,Washington,53033,60283,109
1
2021-01-
28,Pierce,Washington,53053,34481,4
26
2021-01-
28,Snohomish,Washington,53061,2903
8,338
EOF
```

단계 3: 학생 데이터 파일 생성（student.txt）

步骤 3：创建学生数据文件（student.txt）

```
cat > /usr/datas/unit6/student.txt
<< 'EOF'
95001,李勇,男,20,CS
95002,刘晨,女,19,IS
95003,王敏,女,22,MA
95004,张立,男,19,IS
95005,刘刚,男,18,MA
95006,孙庆,男,23,CS
95007,易思玲,女,19,MA
95008,李娜,女,18,CS
```

```
cat > /usr/datas/unit6/student.txt
<< 'EOF'
95001,李勇,男,20,CS
95002,刘晨,女,19,IS
95003,王敏,女,22,MA
95004,张立,男,19,IS
95005,刘刚,男,18,MA
95006,孙庆,男,23,CS
95007,易思玲,女,19,MA
95008,李娜,女,18,CS
```

```
95009,梦圆圆,女,18,MA
95010,孔小涛,男,19,CS
95011,包小柏,男,18,MA
95012,孙花,女,20,CS
95013,冯伟,男,21,CS
95014,王小丽,女,19,CS
95015,王君,男,18,MA
95016,钱国,男,21,MA
95017,王风娟,女,18,IS
95018,王一,女,19,IS
95019,邢小丽,女,19,IS
95020,赵钱,男,21,IS
95021,周二,男,17,MA
95022,郑明,男,20,MA
EOF
```

```
95009,梦圆圆,女,18,MA
95010,孔小涛,男,19,CS
95011,包小柏,男,18,MA
95012,孙花,女,20,CS
95013,冯伟,男,21,CS
95014,王小丽,女,19,CS
95015,王君,男,18,MA
95016,钱国,男,21,MA
95017,王风娟,女,18,IS
95018,王一,女,19,IS
95019,邢小丽,女,19,IS
95020,赵钱,男,21,IS
95021,周二,男,17,MA
95022,郑明,男,20,MA
EOF
```

단계 4: 직원 데이터 파일 생성（employee.txt）

步骤 4：创建员工数据文件（employee.txt）

```
cat > /usr/datas/unit6/employee.txt << 'EOF'
1201,gopal,manager,50000,TP
1202,manisha,cto,50000,TP
1203,khalil,dev,30000,AC
1204,prasanth,dev,30000,AC
1206,kranthi,admin,20000,TP
EOF
```

```
cat > /usr/datas/unit6/employee.txt << 'EOF'
1201,gopal,manager,50000,TP
1202,manisha,cto,50000,TP
1203,khalil,dev,30000,AC
1204,prasanth,dev,30000,AC
1206,kranthi,admin,20000,TP
EOF
```

단계 5: 직원 주소 데이터 파일 생성

步骤 5：创建员工住址数据文件

（employee_address.txt）**

```
cat >
/usr/datas/unit6/employee_address.
txt << 'EOF'
1201,288A,vgiri,jubilee
1202,108I,aoc,ny
1204,144Z,pgutta,hyd
1206,78B,old city,la
1207,720X,hitec,ny
EOF
```

（employee_address.txt）**

```
cat >
/usr/datas/unit6/employee_address.
txt << 'EOF'
1201,288A,vgiri,jubilee
1202,108I,aoc,ny
1204,144Z,pgutta,hyd
1206,78B,old city,la
1207,720X,hitec,ny
EOF
```

단계 6: 직원 연락처 데이터 파일 생성（employee_connection.txt）

```
cat >
/usr/datas/unit6/employee_connecti
on.txt << 'EOF'
1201,2356742,gopal@tp.com
1203,1661663,manisha@tp.com
1204,8887776,khalil@ac.com
1205,9988774,prasanth@ac.com
1206,1231231,kranthi@tp.com
EOF
```

步骤 6：创建员工联系方式数据文件（employee_connection.txt）

```
cat >
/usr/datas/unit6/employee_connecti
on.txt << 'EOF'
1201,2356742,gopal@tp.com
1203,1661663,manisha@tp.com
1204,8887776,khalil@ac.com
1205,9988774,prasanth@ac.com
1206,1231231,kranthi@tp.com
EOF
```

【검증】 데이터 파일이 정상적으로 생성되었는지 확인:

【验证】 查看数据文件是否创建成功：

```
ls -la /usr/datas/unit6/
```

```
ls -la /usr/datas/unit6/
```

예상 결과: txt 파일 5 개가 보여야 합니다.

预期结果：应看到 5 个 txt 文件。

1.2 데이터베이스 및 테이블 구조 생성

1.2 创建数据库与表结构

> **작업 설명**: Beeline 을 실행하고 다음 SQL 문을 실행합니다.

> **操作说明**：启动 Beeline 并执行以下 SQL 语句。

Beeline 실행:

启动 Beeline：

```
hive-bee
```

```
hive-bee
```

단계 1: 데이터베이스 생성 및 사용

步骤 1：创建并使用数据库

```
-- 데이터베이스 생성 (이미 존재하면 건너뜀)
CREATE DATABASE IF NOT EXISTS unit6db;
SHOW DATABASES;
```

```
-- 创建数据库（如已存在则跳过）
CREATE DATABASE IF NOT EXISTS unit6db;
SHOW DATABASES;
```

```
-- unit6db 데이터베이스로 전환
USE unit6db;
SHOW TABLES;
```

```
-- 切换到 unit6db 数据库
USE unit6db;
SHOW TABLES;
```

> ** 📝 결과 기록**: 현재 데이터베이스에 어떤 테이블이 있습니까?
>
> 답: _____

> ** 📝 记录结果**：当前数据库中已有哪些表？
>
> 答：_____

단계 2: 미국 코로나 19 일반 테이블 t_usa_covid19 생성

步骤 2：创建美国新冠普通表 t_usa_covid19

```
-- 테이블이 이미 존재하면 먼저 삭제
DROP TABLE IF EXISTS
t_usa_covid19;

-- 일반 테이블 생성
CREATE TABLE t_usa_covid19 (
    count_date string,
    county     string,
    state      string,
    fips       int,
    cases      int,
    deaths     int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

```
-- 如表已存在，先删除
DROP TABLE IF EXISTS
t_usa_covid19;

-- 创建普通表
CREATE TABLE t_usa_covid19 (
    count_date string,
    county     string,
    state      string,
    fips       int,
    cases      int,
    deaths     int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

```
SHOW TABLES;
```

```
SHOW TABLES;
```

단계 3: t_usa_covid19 에 데이터 로드

步骤 3：加载数据到 t_usa_covid19

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit6/us_covid19.txt'
INTO TABLE t_usa_covid19;

-- 데이터 로드 검증
SELECT * FROM t_usa_covid19 LIMIT
10;
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit6/us_covid19.txt'
INTO TABLE t_usa_covid19;

-- 验证数据加载
SELECT * FROM t_usa_covid19 LIMIT
10;
```

> ** 📝 결과 기록**: 데이터가 총 몇 행입니까?
>
> 답: _____

> ** 📝 记录结果**：数据共有几行？
>
> 答：_____

단계 4: 파티션 테이블 t_usa_covid19_p 생성

步骤 4：创建分区表 t_usa_covid19_p

```
-- count_date 날짜와 state 주를
기준으로 파티션
```

```
-- 基于 count_date 日期和 state 州进行
分区
```

```
CREATE TABLE IF NOT EXISTS
t_usa_covid19_p (
    county string,
    fips   int,
    cases  int,
    deaths int
)
PARTITIONED BY (count_date string,
state string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

SHOW TABLES;
```

```
CREATE TABLE IF NOT EXISTS
t_usa_covid19_p (
    county string,
    fips   int,
    cases  int,
    deaths int
)
PARTITIONED BY (count_date string,
state string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

SHOW TABLES;
```

**단계 5: 파티션 테이블에 데이터 삽입 (동적 파티션) **

步骤 5：将数据导入分区表（动态分区）

```
-- 동적 파티션 모드 활성화
SET
hive.exec.dynamic.partition.mode =
nonstrict;

-- 동적 파티션 삽입
INSERT INTO TABLE t_usa_covid19_p
PARTITION (count_date, state)
SELECT county, fips, cases,
deaths, count_date, state
FROM t_usa_covid19;

-- 파티션 데이터 검증
SELECT * FROM t_usa_covid19_p
LIMIT 10;
SHOW PARTITIONS t_usa_covid19_p;
```

```
-- 开启动态分区模式
SET
hive.exec.dynamic.partition.mode =
nonstrict;

-- 动态分区插入
INSERT INTO TABLE t_usa_covid19_p
PARTITION (count_date, state)
SELECT county, fips, cases,
deaths, count_date, state
FROM t_usa_covid19;

-- 验证分区数据
SELECT * FROM t_usa_covid19_p
LIMIT 10;
SHOW PARTITIONS t_usa_covid19_p;
```

> ** 📝 결과 기록**: 파티션이 총 몇 개입니까?

>

> 답: _____

단계 6: 학생 테이블 student 생성

```
DROP TABLE IF EXISTS student;

CREATE TABLE student (
    num  int,
    name string,
    sex  string,
    age  int,
    dept string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH
'/usr/datas/unit6/student.txt'
OVERWRITE INTO TABLE student;

SELECT * FROM student;
```

> ** 📝 결과 기록**: student 테이블에 레코드가 몇 건입니까?

> ** 📝 记录结果**：共有多少个分区？

>

> 答：_____

步骤 6：创建学生表 student

```
DROP TABLE IF EXISTS student;

CREATE TABLE student (
    num  int,
    name string,
    sex  string,
    age  int,
    dept string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH
'/usr/datas/unit6/student.txt'
OVERWRITE INTO TABLE student;

SELECT * FROM student;
```

> ** 📝 记录结果**：student 表共有多少条记录？

>

> 답: _____

단계 7: 직원 관련 테이블 생성

```
-- 테이블 1: 직원 기본 정보 테이블
DROP TABLE IF EXISTS employee;
CREATE TABLE employee (
    id     int,
    name   string,
    deg    string,
    salary int,
    dept   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH
'/usr/datas/unit6/employee.txt'
INTO TABLE employee;

-- 테이블 2: 직원 주소 정보 테이블
DROP TABLE IF EXISTS
employee_address;
CREATE TABLE employee_address (
    id     int,
    hno    string,
    street string,
    city   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

>

> 答：_____

步骤 7：创建员工相关表

```
-- 表 1：员工基本信息表
DROP TABLE IF EXISTS employee;
CREATE TABLE employee (
    id     int,
    name   string,
    deg    string,
    salary int,
    dept   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH
'/usr/datas/unit6/employee.txt'
INTO TABLE employee;

-- 表 2：员工住址信息表
DROP TABLE IF EXISTS
employee_address;
CREATE TABLE employee_address (
    id     int,
    hno    string,
    street string,
    city   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit6/employee_address
.txt'
INTO TABLE employee_address;

-- 테이블 3: 직원 연락처 테이블
DROP TABLE IF EXISTS
employee_connection;
CREATE TABLE employee_connection (
    id    int,
    phno  string,
    email string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH
'/usr/datas/unit6/employee_connect
ion.txt'
INTO TABLE employee_connection;

-- 세 테이블 데이터 확인
SELECT * FROM employee;
SELECT * FROM employee_address;
SELECT * FROM employee_connection;
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit6/employee_address
.txt'
INTO TABLE employee_address;

-- 表 3：员工联系方式表
DROP TABLE IF EXISTS
employee_connection;
CREATE TABLE employee_connection (
    id    int,
    phno  string,
    email string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH
'/usr/datas/unit6/employee_connect
ion.txt'
INTO TABLE employee_connection;

-- 验证三张表数据
SELECT * FROM employee;
SELECT * FROM employee_address;
SELECT * FROM employee_connection;
```

> ** 📝 사고 문제**: 세 직원 테이블의 id 필드에는 어떤 차이점이 있습니까? JOIN 연산에 어떤 영향을 미칩니까?

>

> 답: _____

> ** 📝 思考题**：三张员工表的 id 字段存在哪些差异？这对 JOIN 操作有什么影响？

>

> 答：_____

☑ 제 1 부 완료 체크리스트

- [ ] 데이터 파일 5 개가 /usr/datas/unit6/ 디렉토리에 정상적으로 생성됨
- [ ] t_usa_covid19 일반 테이블 생성 및 데이터 로드 성공
- [ ] t_usa_covid19_p 파티션 테이블 생성 및 데이터 로드 성공
- [ ] student 테이블 생성 및 데이터 로드 성공
- [ ] employee / employee_address / employee_connection 세 테이블 생성 성공

제 2 부: SELECT 기초 조회 (30 분)

학습 목표

- select_expr 의 다양한 작성 방법 습득
- ALL 과 DISTINCT 의 차이 이해
- 정규표현식을 사용한 열 선택 방법 습득

☑ 第一部分完成检查清单

- [ ] 5 个数据文件已成功创建在 /usr/datas/unit6/ 目录
- [ ] t_usa_covid19 普通表创建并加载数据成功
- [ ] t_usa_covid19_p 分区表创建并加载数据成功
- [ ] student 表创建并加载数据成功
- [ ] employee / employee_address / employee_connection 三张表创建成功

第二部分：SELECT 基础查询（30 分钟）

学习目标

- 掌握 select_expr 的多种写法
- 理解 ALL 与 DISTINCT 的区别
- 能使用正则表达式选择列

2.1 select_expr 선택 표현식

**연습 1: t_usa_covid19_p 테이블의 모든 필드 조회

```
-- 완전한 SQL 문을 작성하세요
SELECT _____ FROM t_usa_covid19_p;
```

> 📝 조회 결과 행 수 기록: _____

**연습 2: t_usa_covid19_p 테이블에서 county（카운티）, cases（확진 수）, deaths（사망 수） 세 필드 조회

```
-- 완전한 SQL 문을 작성하세요
SELECT _____, _____, _____ FROM t_usa_covid19_p;
```

2.1 select_expr 选择表达式

练习 1：查询 t_usa_covid19_p 表中所有字段

```
-- 请填写完整的 SQL 语句
SELECT _____ FROM t_usa_covid19_p;
```

> 📝 记录查询结果的行数：_____

练习 2：查询 t_usa_covid19_p 表中 county（县）、cases（确诊数）、deaths（死亡数）三个字段

```
-- 请填写完整的 SQL 语句
SELECT _____, _____, _____ FROM t_usa_covid19_p;
```

연습 3: 정규표현식을 사용하여 알파벳 'c'로 시작하는 모든 필드 조회

> ** 💡 힌트**: 먼저 `SET hive.support.quoted.identifiers = none;` 을 설정한 후, 백틱 + 정규표현식을 사용합니다.

```
-- 1 단계: 파라미터 설정
SET hive.support.quoted.identifiers = none;

-- 2 단계: 조회문 작성 (c 로 시작하는 모든 열 조회)
SELECT `^c.*` FROM t_usa_covid19_p LIMIT 5;
```

> ** 📝 조회 결과에서 반환된 필드는 무엇입니까?**

>

> 답: _____

练习 3：使用正则表达式查询所有以字母 'c' 开头的字段

> ** 💡 提示**：需要先设置 `SET hive.support.quoted.identifiers = none;`，然后使用反引号+正则表达式

```
-- 第一步：设置参数
SET hive.support.quoted.identifiers = none;

-- 第二步：填写查询语句（查询所有 c 开头的列）
SELECT `^c.*` FROM t_usa_covid19_p LIMIT 5;
```

> ** 📝 查询结果返回了哪些字段？**

>

> 答：_____

연습 4: 현재 사용 중인 데이터베이스 이름 조회

练习 4：查询当前使用的数据库名称

```
-- 완전한 SQL 문을 작성하세요 (FROM 절 불필요)
SELECT _____;
```

```
-- 请填写完整的 SQL 语句（无需 FROM 子句）
SELECT _____;
```

> ** 📝 결과**: _____

> ** 📝 结果是**：_____

연습 5: t_usa_covid19_p 테이블의 county (카운티) 총 개수 통계

练习 5：统计 t_usa_covid19_p 表中 county（县）的总数量

```
SELECT count(county) FROM t_usa_covid19_p;
```

```
SELECT count(county) FROM t_usa_covid19_p;
```

> ** 📝 결과**: _____

> ** 📝 结果是**：_____

2.2 ALL 과 DISTINCT 중복 제거

2.2 ALL 与 DISTINCT 去重

연습 6: t_usa_covid19_p 테이블의 모든 state (주) 조회 (중복 포함)

```
SELECT state FROM t_usa_covid19_p;
```

> ** 조회 결과 행 수**: _____

练习 6：查询 t_usa_covid19_p 表中所有的 state（州），包含重复值

```
SELECT state FROM t_usa_covid19_p;
```

> ** 查询返回了多少行？**：_____

연습 7: ALL 키워드를 사용하여 조회 (연습 6 과 동일한 효과)

```
SELECT ALL state FROM
t_usa_covid19_p;
```

> ** 두 조회 결과의 행 수가 일치합니까?** 예 / 아니오

练习 7：使用 ALL 关键字查询（效果与练习 6 相同）

```
SELECT ALL state FROM
t_usa_covid19_p;
```

> ** 两次查询结果行数是否一致？** 是 / 否

연습 8: DISTINCT 를 사용하여 중복

练习 8：使用 DISTINCT 去重，查询有

제거 후 서로 다른 주 조회

哪些不同的州

```
SELECT DISTINCT state FROM
t_usa_covid19_p;
```

```
SELECT DISTINCT state FROM
t_usa_covid19_p;
```

> ** 📝 중복 제거 후 서로 다른 주는 몇 개입니까?**: _____

> ** 📝 去重后共有多少个不同的州？** : _____

연습 9: DISTINCT 를 사용하여 여러 필드 조합 중복 제거（county + state）**

练习 9：使用 DISTINCT 对多个字段组合去重（county + state）**

```
SELECT DISTINCT county, state FROM
t_usa_covid19_p;
```

```
SELECT DISTINCT county, state FROM
t_usa_covid19_p;
```

> ** 📝 사고 문제**: 다중 필드 DISTINCT 의 중복 제거 논리는 무엇입니까? 단일 필드와 어떻게 다릅니까?

>

> 답: _____

> ** 📝 思考题**：多字段 DISTINCT 的去重逻辑是什么？与单字段有何不同？

>

> 答：_____

연습 10（심화）: 학생 테이블에서 서로 다른 학과（dept） 수 통계

```
-- SQL 문을 직접 작성하세요
```

> ** 결과**: _____

☑ 제 2 부 소결

문법	설명	비고
`SELECT *`	모든 필드 조회	
`SELECT col1, col2`	지정 필드 조회	
`` SELECT `^c.*` ``	정규표현식으로 열 선택	먼저 파라미터 설정 필요
`SELECT ALL`	전체 행 반환（중복 포함）	기본 동작
`SELECT DISTINCT`	중복 행 제거	

练习 10（扩展）：统计学生表中有多少个不同的系别（dept）

```
-- 请自行编写 SQL 语句
```

> ** 结果**：_____

☑ 第二部分小结

语法	说明	示例
`SELECT *`	查询所有字段	
`SELECT col1, col2`	查询指定字段	
`` SELECT `^c.*` ``	正则表达式选列	需先设置参数
`SELECT ALL`	返回全部行（含重复）	默认行为
`SELECT DISTINCT`	去除重复行	

제 3 부: WHERE 절과 파티션 프루닝 (25 분)

학습 목표

- WHERE 절의 다양한 조건 표현식 습득
- WHERE 절에서 집계 함수를 사용할 수 없는 이유 이해
- 파티션 프루닝의 개념과 사용 방법 습득

3.1 WHERE 기본 조건

**연습 1: 불리언 표현식 테스트 — 조건이 항상 false 일 때의 결과

```
SELECT * FROM t_usa_covid19_p
WHERE 1 > 2;
```

第三部分：WHERE 子句与分区裁剪（25 分钟）

学习目标

- 掌握 WHERE 子句的各种条件表达式
- 理解为什么 WHERE 中不能使用聚合函数
- 掌握分区裁剪的概念和用法

3.1 WHERE 基础条件

练习 1：布尔表达式测试——查询条件恒为 false 时的结果

```
SELECT * FROM t_usa_covid19_p
WHERE 1 > 2;
```

> ** 📝 조회 결과가 몇 행입니까? 이유는 무엇입니까?**

>

> 답: _____

연습 2: 조건이 항상 true 일 때의 결과

```
SELECT * FROM t_usa_covid19_p
WHERE 1 = 1 LIMIT 5;
```

> ** 📝 이 두 연습을 통해 WHERE 절의 본질적인 특성은 무엇임을 알 수 있습니까?**

>

> 답: _____

연습 3: 함수를 WHERE 조건으로 사용 — 주 이름의 알파벳 길이가 10 자를 초과하는 레코드 찾기

> ** 📝 查询返回多少行？为什么？**

>

> 答：_____

练习 2：查询条件恒为 true 时的结果

```
SELECT * FROM t_usa_covid19_p
WHERE 1 = 1 LIMIT 5;
```

> ** 📝 这两个练习说明了 WHERE 子句的什么本质？**

>

> 答：_____

练习 3：使用函数作为 WHERE 条件——找出州名字母长度超过 10 位的记录

```
SELECT * FROM t_usa_covid19_p
WHERE length(state) > 10;
```

> ** 📝 조건에 해당하는 주는 어디입니까?**

>

> 답: _____

연습 4: WHERE 절의 서브쿼리 — 특정 목록에 있는 county 레코드 찾기**

```
-- county 가 'Los Angeles' 또는 'Cook'인 레코드 조회
SELECT * FROM t_usa_covid19
WHERE county IN ('Los Angeles', 'Cook');
```

> ** 📝 어떤 레코드가 반환됩니까?**

>

> 답: _____

```
SELECT * FROM t_usa_covid19_p
WHERE length(state) > 10;
```

> ** 📝 符合条件的州有哪些？**

>

> 答：_____

练习 4：WHERE 子句中的子查询——找出 county 在某个特定列表中的记录**

```
-- 查询 county 为'Los Angeles'或'Cook'的记录
SELECT * FROM t_usa_covid19
WHERE county IN ('Los Angeles', 'Cook');
```

> ** 📝 结果返回了哪些记录？**

>

> 答：_____

연습 5（오류 예시）: WHERE 절에서 집계 함수 사용 시도（오류 메시지 확인）

练习 5（错误示范）：尝试在 WHERE 子句中使用聚合函数（观察错误信息）

```
-- 주의: 이 SQL 은 오류가 발생합니다. 오류 메시지를 주의 깊게 읽으세요
SELECT state, count(deaths)
FROM t_usa_covid19_p
WHERE count(deaths) > 100
GROUP BY state;
```

```
-- 注意：这条 SQL 会报错，请仔细阅读错误信息
SELECT state, count(deaths)
FROM t_usa_covid19_p
WHERE count(deaths) > 100
GROUP BY state;
```

> ** 📝 오류 메시지의 핵심 내용 기록**:
>
> 답: _____

> ** 📝 记录错误信息的关键内容**：
>
> 答：_____

> ** 📝 올바른 작성법**（집계 조건을 HAVING 으로 변경）：
>
> ```sql
> SELECT state, count(deaths)
> FROM t_usa_covid19_p
> WHERE count_date = "2021-01-28"
> GROUP BY state
```

> ** 📝 正确的写法应该是**（请将聚合条件改用 HAVING）：
>
> ```sql
> SELECT state, count(deaths)
> FROM t_usa_covid19_p
> WHERE count_date = "2021-01-28"
> GROUP BY state
```

> HAVING _____;

> ```


3.2 파티션 프루닝 (Partition Pruning)

> **💡 개념 설명**: 파티션 프루닝이란 파티션 테이블을 조회할 때, WHERE 절의 파티션 필드 필터를 통해 Hive 가 전체 테이블을 스캔하지 않고 조건에 맞는 파티션만 스캔하도록 하여 조회 성능을 향상시키는 것입니다.

연습 6: 파티션 테이블 전체 스캔 (파티션 프루닝 없음)

```
-- deaths > 1000 인 모든 레코드 조회

SELECT * FROM t_usa_covid19_p
WHERE deaths > 1000;
```

> **📝 결과 행 수 기록**: _____

> HAVING _____;

> ```


3.2 分区裁剪（Partition Pruning）

> **💡 概念说明**：分区裁剪是指在查询分区表时，通过 WHERE 子句中对分区字段的过滤，让 Hive 只扫描满足条件的分区，而不是全表扫描，从而提高查询效率。

练习 6：对分区表进行全表扫描（无分区裁剪）

```
-- 查询所有加州(California)的记录, deaths > 1000
SELECT * FROM t_usa_covid19_p
WHERE deaths > 1000;
```

> **📝 记录结果行数**：_____

연습 7: 단일 파티션 프루닝 사용 (state 는 파티션 필드)

```
-- state 는 파티션 필드로, 파티션 프루닝 가능
SELECT * FROM t_usa_covid19_p
WHERE state = 'California'
AND deaths > 1000;
```

> ** 📝 결과 행 수**: _____

>

> ** 📝 연습 6 과 비교하여 파티션 프루닝의 장점은 무엇입니까?**

>

> 답: _____

练习 8：多分区裁剪（同时指定 count_date 和 state 两个分区字段）

练习 7：使用单分区裁剪（state 是分区字段）

```
-- state 是分区字段，可以进行分区裁剪
SELECT * FROM t_usa_covid19_p
WHERE state = 'California'
AND deaths > 1000;
```

> ** 📝 结果行数**：_____

>

> ** 📝 与练习 6 相比，分区裁剪有什么优势？**

>

> 答：_____

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
AND deaths > 1000;
```

> ** 📝 결과 행 수**: _____

> ** 📝 사고 문제**: 어떤 필드가 파티션 필드입니까? 조회 시 WHERE 절에서 어떤 유형의 필드를 우선적으로 사용해야 합니까?

>

> 답: _____

연습 9（심화）: student 테이블에서 WHERE 를 사용하여 컴퓨터학과（CS）의 19 세 초과 남학생 조회

```
-- SQL 문을 직접 작성하세요
```

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
AND deaths > 1000;
```

> ** 📝 结果行数** : _____

> ** 📝 思考题** : 哪些字段是分区字段？查询时应优先在 WHERE 中使用哪类字段？

>

> 答 : _____

练习 9（扩展） : 在 student 表中，使用 WHERE 查询计算机系（CS）中年龄大于 19 岁的男生

```
-- 请自行编写 SQL 语句
```

> ** 📝 조건에 해당하는 학생은 누구입니까?**

>

> 답: _____

제 4 부: GROUP BY 와 HAVING (30 분)

학습 목표

- GROUP BY 의 작동 원리와 문법 제한 이해
- 다양한 집계 함수 사용 방법 습득
- HAVING 과 WHERE 의 핵심 차이점 이해

4.1 GROUP BY 기초

연습 1: 주 (state) 별로 그룹화하여 각 주의 카운티 수 통계

> ** 📝 符合条件的学生有哪些？**

>

> 答：_____

第四部分：GROUP BY 与 HAVING（30 分钟）

学习目标

- 理解 GROUP BY 的工作原理和语法限制
- 掌握各种聚合函数的使用
- 理解 HAVING 与 WHERE 的核心区别

4.1 GROUP BY 基础

练习 1：按州(state)分组，统计每个州的县数量

```
SELECT state, count(county) AS
county_count
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

```
SELECT state, count(county) AS
county_count
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

> ** 📝 조회 결과 기록**（표에 입력）：

>

> | 주 이름 | 카운티 수 |

> |--------|---------|

> | | |

> | | |

연습 2（오류 예시）: SELECT 에 비집계·비그룹화 필드 포함（오류 확인）

```
-- 이 SQL 은 오류 발생: deaths 는
그룹화 필드가 아니고 집계 함수로도
처리되지 않음
SELECT state, deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

> ** 📝 记录查询结果**（填入表格）：

>

> | 州名 | 县数量 |

> |------|--------|

> | | |

> | | |

练习 2（错误示范）：将非聚合、非分组字段放入 SELECT 中（观察错误）

```
-- 这条 SQL 会报错：deaths 不是分组字
段，也没有被聚合函数处理

SELECT state, deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

> ** 📝 오류 메시지의 핵심 키워드 기록**: _____

> ** 📝 记录错误信息的关键词** : _____

> ** 📝 오류를 통해 GROUP BY 의 문법 제한 규칙을 설명하세요**:
>
> 답: _____

> ** 📝 根据错误，说明 GROUP BY 的语法限制规则** :
>
> 答 : _____

연습 3: 올바른 작성법 — deaths 에 집계 함수 적용**

练习 3：正确写法——将 deaths 用聚合函数处理**

```
-- 주별 그룹화, 각 주의 총 사망자 수 통계
SELECT state, sum(deaths) AS total_deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

```
-- 按州分组，统计每个州的总死亡人数

SELECT state, sum(deaths) AS total_deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

> ** 📝 사망자 수가 가장 많은 주는 어디입니까?**: _____

> ** 📝 死亡人数最多的州是哪个？** : _____

연습 4: 다중 집계 함수 조합 사용

```
-- 주별 그룹화, 동시에 계산: 총 확진 수, 총 사망 수, 평균 확진 수, 최대 확진 수
SELECT
    state,
    sum(cases)  AS total_cases,
    sum(deaths) AS total_deaths,
    round(avg(cases), 2) AS avg_cases,
    max(cases)  AS max_cases
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

> ** 📝 round() 함수의 역할은 무엇입니까?**

>

> 답: _____

연습 5（심화）: student 테이블에서 성별과 학과로 이중 그룹화하여 각 그룹의

练习 4：多聚合函数组合使用

```
-- 按州分组，同时计算：确诊总数、死亡总数、平均确诊数、最大确诊数

SELECT
    state,
    sum(cases)  AS total_cases,
    sum(deaths) AS total_deaths,
    round(avg(cases), 2) AS avg_cases,
    max(cases)  AS max_cases
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

> ** 📝 round()函数的作用是什么？**

>

> 答：_____

练习 5（扩展）：对 student 表，按性别和系别进行双字段分组，统计每组的人数

인원 수와 평균 연령 통계

和平均年龄

```
-- SQL 문을 직접 작성하세요
```

```
-- 请自行编写 SQL 语句
```

4.2 HAVING 절

4.2 HAVING 子句

> ** 💡 HAVING vs WHERE 비교**:

> ** 💡 HAVING vs WHERE 对比**：

> - WHERE: 그룹화 **이전에** 행을 필터링, 집계 함수 사용 불가

> - WHERE：在分组**之前**过滤行，不能使用聚合函数

> - HAVING: 그룹화 **이후에** 그룹을 필터링, 집계 함수 사용 가능

> - HAVING：在分组**之后**过滤组，可以使用聚合函数

연습 6（오류 작성법）: WHERE 에서 집계 함수를 사용하여 그룹화 결과 필터링 시도**

练习 6（错误写法）：尝试在 WHERE 中使用聚合函数过滤分组后的结果

```
-- 이 SQL 은 오류 발생
SELECT state, sum(deaths)
FROM t_usa_covid19_p
```

```
-- 这条 SQL 会报错
SELECT state, sum(deaths)
FROM t_usa_covid19_p
```

```
WHERE count_date = "2021-01-28"
AND sum(deaths) > 10000
GROUP BY state;
```

> ** 📝 이 오류의 원인은 무엇입니까?**
>
> 답: _____

연습 7: HAVING 을 올바르게 사용하여 필터링 — 총 사망자 수가 10,000 을 초과하는 주 찾기

```
SELECT state, sum(deaths) AS
total_deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING sum(deaths) > 10000;
```

> ** 📝 조건에 해당하는 주는 어디입니까?**
>
> 답: _____

```
WHERE count_date = "2021-01-28"
AND sum(deaths) > 10000
GROUP BY state;
```

> ** 📝 此错误的原因是什么？**
>
> 答：_____

练习 7：使用 HAVING 正确过滤——找出总死亡人数超过 10000 的州

```
SELECT state, sum(deaths) AS
total_deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING sum(deaths) > 10000;
```

> ** 📝 符合条件的州有哪些？**
>
> 答：_____

연습 8（최적화 작성법）:
HAVING 에서 별칭을 사용하여 집계 결과 참조

```
-- 더 효율적인 작성법: having 에서 별칭을 직접 참조하여 중복 계산 방지
SELECT state, sum(deaths) AS cnts
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING cnts > 10000;
```

> ** 📝 연습 7 과 비교하여 이 작성법의 장점은 무엇입니까?**
>
> 답: _____

练习 8（优化写法）：使用别名在 HAVING 中引用聚合结果

```
-- 更高效的写法：having 直接引用别名，避免重复计算
SELECT state, sum(deaths) AS cnts
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING cnts > 10000;
```

> ** 📝 与练习 7 相比，此写法有什么优势？**
>
> 答：_____

연습 9（종합）: 컴퓨터학과(CS) 또는 정보학과(IS) 중, 평균 연령이 19 세를 초과하고 인원이 3 명보다 많은 학과 찾기

练习 9（综合）：找出计算机系(CS)或信息系(IS)中，平均年龄超过 19 岁且人数大于 3 人的系别

```
-- WHERE + GROUP BY + HAVING 을 조합하여 SQL 문을 직접 작성하세요
```

```
-- 请自行编写 SQL 语句，综合使用 WHERE + GROUP BY + HAVING
```

☑ WHERE 와 HAVING 비교 정리

☑ WHERE 与 HAVING 对比总结

| 특성 | WHERE | HAVING |

|------|-------|--------|

| 실행 시점 | 그룹화 **이전** | 그룹화 **이후** |

| 집계 함수 사용 가능 여부 | **불가** | **가능** |

| 필터 대상 | 행（row） | 그룹（group） |

| 단독 사용 가능 여부 | 가능 | 일반적으로 GROUP BY 와 함께 사용 |

| 特性 | WHERE | HAVING |

|------|-------|--------|

| 执行时机 | 分组**之前** | 分组**之后** |

| 能否使用聚合函数 | **不能** | **能** |

| 过滤对象 | 行（row） | 组（group） |

| 是否可单独使用 | 是 | 一般配合 GROUP BY |

제 5 부: LIMIT 와 SQL 실행 순서

第五部分：LIMIT 与 SQL 执行顺序（20

(20 분)

학습 목표

- LIMIT 절의 두 가지 파라미터 형식 습득

- SQL 의 전체 실행 순서 이해 및 암기

5.1 LIMIT 절

> ** 💡 문법 설명**:

> - `LIMIT n`: 처음 n 행 반환

> - `LIMIT offset, n`: offset 행부터 시작하여 (오프셋은 0 부터 계산) n 행 반환

연습 1: 캘리포니아 (California) 2021-01-28 의 모든 레코드 조회 (제한 없음)

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California";
```

分钟)

学习目标

- 掌握 LIMIT 子句的两种参数形式

- 理解并记忆 SQL 的完整执行顺序

5.1 LIMIT 子句

> ** 💡 语法说明** :

> - `LIMIT n`：返回前 n 行

> - `LIMIT offset, n`：从第 offset 行开始（偏移量从 0 计），返回 n 行

练习 1 : 查询加州（California）2021-01-28 的所有记录（无限制）

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California";
```

> ** 📝 총 몇 행이 반환됩니까?**: _____

연습 2: LIMIT 단일 파라미터 — 처음 5 개 레코드 반환

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
LIMIT 5;
```

연습 3: LIMIT 이중 파라미터 — 2 번째부터 (오프셋 2), 3 개 반환

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
LIMIT 2, 3;
```

> ** 📝 주의**: 첫 번째 파라미터 (2) 는 오프셋으로, 몇 번째 행부터 시작합니까?

> ** 📝 共返回多少行？** ：_____

练习 2 ：LIMIT 单参数——返回前 5 条记录

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
LIMIT 5;
```

练习 3 ：LIMIT 双参数——从第 2 条开始（偏移量为 2），返回 3 条

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
LIMIT 2, 3;
```

> ** 📝 注意**：第一个参数（2）是偏移量，从第几行开始（0 表示第 1 行）？

（0 은 첫 번째 행을 의미）?

>

> 답: _____

연습 4（심화）: student 테이블에서 3 번째부터 5 번째 학생 조회（num 정렬 후 3~5 번째 레코드）

```
-- 힌트: 먼저 ORDER BY num 으로 정렬 후, LIMIT 으로 오프셋 지정
SELECT * FROM student
ORDER BY num
LIMIT _____, _____;
```

5.2 SQL 실행 순서 분석

> **💡 핵심 지식**: Hive SQL 의 실행 순서（순서대로）:

>

> **FROM → WHERE → GROUP BY（집계

>

> 答：_____

练习 4（扩展）：查询 student 表中第 3 名到第 5 名的学生（按 num 排序后取第 3~5 条）

```
-- 提示：先用 ORDER BY num 排序，再用 LIMIT 取偏移
SELECT * FROM student
ORDER BY num
LIMIT _____, _____;
```

5.2 SQL 执行顺序分析

> **💡 核心知识点**：Hive SQL 的执行顺序（从先到后）：

>

> **FROM → WHERE → GROUP BY（含聚合函数）→ HAVING → SELECT → ORDER BY →

함수 포함) → HAVING → SELECT → ORDER BY → LIMIT**

연습 5: 다음 SQL 문의 실행 단계 분석

```
SELECT state, sum(deaths) AS cnts
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING cnts > 10000;
```

> ** 실행 순서에 따라 각 단계가 무엇을 수행하는지 분석하세요**:

>

> 1. FROM:
_____

> 2. WHERE:
_____

> 3. GROUP BY + sum():
_____

> 4. HAVING:
_____

> 5. SELECT:
_____

LIMIT**

练习 5：分析以下 SQL 语句的执行步骤

```
SELECT state, sum(deaths) AS cnts
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING cnts > 10000;
```

> ** 请按执行顺序分析每一步做了什么**：

>

> 1. FROM：
_____

> 2. WHERE：
_____

> 3. GROUP BY + sum()：
_____

> 4. HAVING：
_____

> 5. SELECT：
_____

연습 6（빈칸 채우기）: 실행 순서에 근거하여 다음 중 어떤 작성법이 올바른지 판단

（） A. SELECT state, sum(deaths) FROM t_usa_covid19_p WHERE sum(deaths) > 100 GROUP BY state;

（） B. SELECT state, sum(deaths) FROM t_usa_covid19_p GROUP BY state HAVING sum(deaths) > 100;

（） C. SELECT state FROM t_usa_covid19_p GROUP BY state WHERE state = 'Texas';

（） D. SELECT state, count(*) AS cnt FROM t_usa_covid19_p GROUP BY state HAVING cnt > 3;

> **올바른 선택지는（복수 선택）**: _____

제 6 부: 고급 정렬 문법（35 분）

학습 목표

- ORDER BY（전역 정렬）의 사용법 습득

练习 6（填空题）：根据执行顺序，判断以下哪些写法正确

（） A. SELECT state, sum(deaths) FROM t_usa_covid19_p WHERE sum(deaths) > 100 GROUP BY state;

（） B. SELECT state, sum(deaths) FROM t_usa_covid19_p GROUP BY state HAVING sum(deaths) > 100;

（） C. SELECT state FROM t_usa_covid19_p GROUP BY state WHERE state = 'Texas';

（） D. SELECT state, count(*) AS cnt FROM t_usa_covid19_p GROUP BY state HAVING cnt > 3;

> **正确的选项是（多选）**：_____

第六部分：高阶排序语法（35 分钟）

学习目标

- 掌握 ORDER BY（全局排序）的用法

- CLUSTER BY 의 작동 원리와 한계 이해

- DISTRIBUTE BY + SORT BY 조합 사용법 습득

- 네 가지 정렬 방식의 관계와 차이점 이해

6.1 ORDER BY 전역 정렬

**연습 1: deaths（사망 수）를 기준으로 캘리포니아 레코드를 오름차순 정렬

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
ORDER BY deaths;
-- 주의: 기본값은 ASC 오름차순, NULL 값은 맨 앞에 배치（NULLS FIRST）
```

**연습 2: deaths 를 내림차순 정렬하고 상위 3 개 레코드 반환（사망자 수가 가장 많은 카운티 3 개）

- 理解 CLUSTER BY 的工作原理与局限性

- 掌握 DISTRIBUTE BY + SORT BY 的组合用法

- 理解四种排序方式的关系和区别

6.1 ORDER BY 全局排序

练习 1：按 deaths（死亡数）升序排序加州的记录

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
ORDER BY deaths;
-- 注意：默认 ASC 升序，NULL 值排在最前面（NULLS FIRST）
```

练习 2：按 deaths 降序排序，并限制返回前 3 条（死亡人数最多的 3 个县）

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
ORDER BY deaths DESC
LIMIT 3;
```

> ** 📝 사망자 수가 가장 많은 카운티 3 개는?**
>
> 답: _____

연습 3: 다중 필드 정렬 — state 를 오름차순으로, deaths 를 내림차순으로

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
ORDER BY state ASC, deaths DESC
LIMIT 10;
```

연습 4（심화）: student 테이블에서

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
ORDER BY deaths DESC
LIMIT 3;
```

> ** 📝 死亡人数最多的 3 个县是？**
>
> 答：_____

练习 3：多字段排序——先按 state 升序，再按 deaths 降序

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
ORDER BY state ASC, deaths DESC
LIMIT 10;
```

练习 4（扩展）：在 student 表中，按

age 를 오름차순 정렬 (NULL 값은 맨 뒤에 배치)

```
SELECT * FROM student
ORDER BY age ASC NULLS LAST;
```

> ** 📝 NULLS LAST 의 의미는 무엇입니까?**
>
> 답: _____

6.2 CLUSTER BY 그룹 정렬

> ** 💡 개념**: CLUSTER BY 는 지정된 필드를 기준으로 해시 그룹화를 수행하고, 각 그룹 내에서 해당 필드를 **오름차순**으로 정렬합니다 (오름차순만 지원). 그룹 수는 reduce task 수에 따라 결정됩니다.

연습 5: reduce task 수를 지정하지 않고, student 테이블을 num 으로

age 升序排序（NULL 值排最后）

```
SELECT * FROM student
ORDER BY age ASC NULLS LAST;
```

> ** 📝 NULLS LAST 的含义是什么？**
>
> 答：_____

6.2 CLUSTER BY 分组排序

> ** 💡 概念**：CLUSTER BY 根据指定字段进行哈希分组，并在每个组内对该字段进行**升序**排序（只能升序）。分组数量取决于 reducetask 的数量。

练习 5：不指定 reduce task 个数，对 student 表按 num 进行 CLUSTER BY

CLUSTER BY

```
-- 기본값: reduce task 수 지정 없음
SELECT * FROM student CLUSTER BY num;
```

> ** 로그에서 어떤 정보가 표시됩니까（Number of reduce tasks...）?**

>

> 답: _____

연습 6: 수동으로 reduce task 를 2 개로 설정하고 CLUSTER BY 재실행**

```
SET mapreduce.job.reduces = 2;
SELECT * FROM student CLUSTER BY num;
```

> ** reduce task 를 2 개로 설정 후, 결과가 몇 그룹으로 나뉩니까?**

>

CLUSTER BY

```
-- 默认不指定 reduce task 个数
SELECT * FROM student CLUSTER BY num;
```

> ** 日志中提示了什么信息（Number of reduce tasks...）？**

>

> 答：_____

练习 6：手动设置 2 个 reduce task，再次执行 CLUSTER BY**

```
SET mapreduce.job.reduces = 2;
SELECT * FROM student CLUSTER BY num;
```

> ** 设置 2 个 reducetask 后，结果分成了几组？**

>

> 답: _____

연습 7（한계 이해）: CLUSTER BY 로 "sex 로 그룹화, 그룹 내 age 내림차순" 구현 시도 — 가능 여부 확인

```
-- 이 작성법 시도（오류 발생）
SELECT * FROM student CLUSTER BY sex ORDER BY age DESC;
```

> ** 📝 오류 메시지는 무엇입니까? CLUSTER BY 가 이 요구사항을 구현할 수 없는 이유는?**

>

> 답: _____

6.3 DISTRIBUTE BY + SORT BY

> ** 💡 개념**:

> - DISTRIBUTE BY: 지정된 필드를 기준으로

> 答：_____

练习 7（理解局限性）：尝试用 CLUSTER BY 实现"按 sex 分组，组内按 age 降序"——验证是否可行

```
-- 尝试这种写法（会报错）
SELECT * FROM student CLUSTER BY sex ORDER BY age DESC;
```

> ** 📝 报错信息是什么？为什么 CLUSTER BY 无法实现此需求？**

>

> 答：_____

6.3 DISTRIBUTE BY + SORT BY

> ** 💡 概念**：

> - DISTRIBUTE BY：负责按指定字段进行

해시 **그룹화**（파티셔닝） 수행

> - SORT BY: 각 그룹 내 **로컬 정렬** 수행

> - 두 필드가 **달라도 됨** — CLUSTER BY 의 한계를 보완

연습 8: sex(성별)로 그룹화하고, 그룹 내에서 age(나이)를 **내림차순** 정렬

```
SELECT * FROM student
DISTRIBUTE BY sex
SORT BY age DESC;
```

> ** 📝 결과 관찰: 남학생 그룹과 여학생 그룹이 분리되었습니까? 각 그룹 내 나이가 내림차순으로 정렬되었습니까?**

>

> 답: _____

연습 9: 동치 관계 확인 — DISTRIBUTE BY 와 SORT BY 필드가 같을 때 CLUSTER BY 와 동일

哈希**分组**（分区）

> - SORT BY：负责每个分组内的**局部排序**

> - 两者字段**可以不同**，弥补了 CLUSTER BY 的局限性

练习 8：按 sex 性别分组，组内按 age 年龄**降序**排列

```
SELECT * FROM student
DISTRIBUTE BY sex
SORT BY age DESC;
```

> ** 📝 观察结果：男生组和女生组是否分开了？每组内年龄是否降序？**

>

> 答：_____

练习 9：验证等价关系——当 DISTRIBUTE BY 和 SORT BY 字段相同时，等价于 CLUSTER BY

```
-- 방식 1: DISTRIBUTE BY + SORT BY (같은 필드)
SELECT * FROM student DISTRIBUTE BY num SORT BY num LIMIT 5;

-- 방식 2: CLUSTER BY
SELECT * FROM student CLUSTER BY num LIMIT 5;
```

```
-- 方式1：DISTRIBUTE BY + SORT BY（字段相同）
SELECT * FROM student DISTRIBUTE BY num SORT BY num LIMIT 5;

-- 方式2：CLUSTER BY
SELECT * FROM student CLUSTER BY num LIMIT 5;
```

> ** 📝 두 방식의 결과가 같습니까? 이것이 의미하는 것은 무엇입니까?**
>
> 답: _____

> ** 📝 两种方式的结果是否相同？这说明了什么？**
>
> 答：_____

☑ 네 가지 정렬 방식 비교 정리

정렬 방식	Reducer 수	그룹화	정렬 범위	내림차순 지원 여부
ORDER BY	1 개	없음	전역	✅
SORT BY	다수	없음	그룹 내 로컬	✅

☑ 四种排序方式对比总结

排序方式	Reducer 数量	分组	排序范围	是否支持降序
ORDER BY	1 个	无	全局	✅
SORT BY	多个	无	组内局部	✅

| DISTRIBUTE BY | 다수 | ✅ | 없음 | — |

| CLUSTER BY | 다수 | ✅ | 그룹 내 로컬 | ❌ (오름차순만) |

제 7 부: UNION 연합 조회와 서브쿼리 (30 분)

학습 목표

- UNION 과 UNION ALL 의 차이 습득

- FROM 절의 서브쿼리 이해

- WHERE 절의 비상관 서브쿼리 (IN/NOT IN) 와 상관 서브쿼리 (EXISTS) 습득

7.1 UNION 연합 조회

연습 1: 두 테스트 테이블 생성, UNION 과 UNION ALL 의 차이 이해

```
-- 테스트 테이블 A 생성
```

| DISTRIBUTE BY | 多个 | ✅ | 无 | — |

| CLUSTER BY | 多个 | ✅ | 组内局部 | ❌ (只能升序) |

第七部分：UNION 联合查询与子查询 (30 分钟)

学习目标

- 掌握 UNION 和 UNION ALL 的区别

- 理解 FROM 子句中的子查询

- 掌握 WHERE 子句中的不相关子查询 (IN/NOT IN) 和相关子查询 (EXISTS)

7.1 UNION 联合查询

练习 1：创建两个测试表，理解 UNION 与 UNION ALL 的差异

```
-- 创建测试表 A
```

```
DROP TABLE IF EXISTS t_union_a;
CREATE TABLE t_union_a (id int,
name string)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

-- 테스트 테이블 B 생성
DROP TABLE IF EXISTS t_union_b;
CREATE TABLE t_union_b (id int,
name string)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

-- 테스트 데이터 삽입
INSERT INTO t_union_a VALUES (1,
'Alice'), (2, 'Bob'), (3,
'Charlie');
INSERT INTO t_union_b VALUES (3,
'Charlie'), (4, 'David'), (5,
'Eve');
```

```
DROP TABLE IF EXISTS t_union_a;
CREATE TABLE t_union_a (id int,
name string)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

-- 创建测试表 B
DROP TABLE IF EXISTS t_union_b;
CREATE TABLE t_union_b (id int,
name string)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

-- 插入测试数据（表 A 中有重复数据）
INSERT INTO t_union_a VALUES (1,
'Alice'), (2, 'Bob'), (3,
'Charlie');
INSERT INTO t_union_b VALUES (3,
'Charlie'), (4, 'David'), (5,
'Eve');
```

연습 2: UNION ALL 사용 (중복 행 유지)

练习 2：使用 UNION ALL（保留重复行）

```
SELECT id, name FROM t_union_a
UNION ALL
SELECT id, name FROM t_union_b;
```

```
SELECT id, name FROM t_union_a
UNION ALL
SELECT id, name FROM t_union_b;
```

> ** 📝 결과는 총 몇 행입니까?**: _____ (두 테이블 행 수의 합)

> ** 📝 结果共几行？**：_____（两表行数之和）

연습 3: UNION 사용 (자동 중복 제거)

```
SELECT id, name FROM t_union_a
UNION
SELECT id, name FROM t_union_b;
```

> ** 📝 결과는 총 몇 행입니까? UNION 과 UNION ALL 의 차이점은 무엇입니까?**
>
> 답: _____

연습 4: 실제 시나리오에서 UNION 활용 — 서로 다른 주의 데이터 합치기

```
-- California 와 Texas 의 코로나 19 데이터 합치기
SELECT county, cases, deaths, state FROM t_usa_covid19_p
WHERE state = 'California'
UNION ALL
SELECT county, cases, deaths, state FROM t_usa_covid19_p
```

练习 3：使用 UNION（自动去重）

```
SELECT id, name FROM t_union_a
UNION
SELECT id, name FROM t_union_b;
```

> ** 📝 结果共几行？UNION 与 UNION ALL 的区别是什么？**
>
> 答：_____

练习 4：UNION 在实际场景中的应用——合并不同州的数据

```
-- 合并 California 和 Texas 的新冠数据
SELECT county, cases, deaths, state FROM t_usa_covid19_p
WHERE state = 'California'
UNION ALL
SELECT county, cases, deaths, state FROM t_usa_covid19_p
```

```
WHERE state = 'Texas';
```

> ** 📝 결과는 총 몇 행입니까?**: _____

7.2 FROM 절의 서브쿼리

> ** 💡 개념**: SELECT 조회 결과를 임시 테이블로 사용하여 FROM 절에 배치합니다. 서브쿼리에는 반드시 별칭이 있어야 합니다.

연습 5: FROM 서브쿼리 사용 — 먼저 student 테이블에서 num 과 name 을 가져온 후, 결과에서 필터링**

```
-- 외부 조회에서 서브쿼리 결과 참조
SELECT num
FROM (
    SELECT num, name FROM student
) tmp
WHERE tmp.name LIKE '王%';
```

> ** 📝 왕(王) 씨 학생의 학번은 어떻게

```
WHERE state = 'Texas';
```

> ** 📝 结果共几行？** : _____

7.2 FROM 子句中的子查询

> ** 💡 概念** : 将一个 SELECT 查询的结果作为临时表，放在 FROM 子句中使用。子查询必须有别名。

练习 5 : 使用 FROM 子查询——先获取 student 表中 num 和 name，再从结果中过滤**

```
-- 外层查询引用子查询结果
SELECT num
FROM (
    SELECT num, name FROM student
) tmp
WHERE tmp.name LIKE '王%';
```

> ** 📝 姓王的同学学号有哪些？**

됩니까?**

>

> 답: _____

연습 6: FROM 서브쿼리와 UNION 결합 — 각 주의 최대 확진 수 레코드 찾기

```
SELECT state, max_cases
FROM (
    SELECT state, max(cases) AS
max_cases
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    GROUP BY state
) t
WHERE max_cases > 100000;
```

> ** 📝 결과에 포함된 주는 어디입니까?**

>

> 답: _____

练习 6：FROM 子查询结合 UNION——找出各州确诊数最多的记录

```
SELECT state, max_cases
FROM (
    SELECT state, max(cases) AS
max_cases
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    GROUP BY state
) t
WHERE max_cases > 100000;
```

> ** 📝 结果中有哪些州？**

>

> 答：_____

7.3 WHERE 절의 서브쿼리

연습 7: 비상관 서브쿼리 (IN) — employee_address 테이블에서 id 가 employee 테이블에 존재하는 레코드 조회

```
-- 비상관 서브쿼리: 서브쿼리 결과를 IN 의 목록으로 사용
SELECT *
FROM employee_address
WHERE id IN (SELECT id FROM employee);
```

> ** 결과가 몇 행 반환됩니까? 어떤 id 가 매칭됩니까?**
>
> 답: _____

연습 8: 비상관 서브쿼리 (NOT IN) — employee_address 테이블에서 id 가 employee 테이블에 없는 레코드 조회

7.3 WHERE 子句中的子查询

练习 7：不相关子查询（IN）——查询 employee_address 表中，id 在 employee 表中存在的记录

```
-- 不相关子查询：子查询结果作为 IN 的列表
SELECT *
FROM employee_address
WHERE id IN (SELECT id FROM employee);
```

> ** 结果返回几行？哪些 id 有匹配？**
>
> 答：_____

练习 8：不相关子查询（NOT IN）——查询 employee_address 表中，id 不在 employee 表中的记录

```
SELECT *
FROM employee_address
WHERE id NOT IN (SELECT id FROM
employee);
```

> ** 📝 어떤 직원 주소 레코드에 해당하는 직원 기본 정보가 없습니까?**
>
> 답: _____

연습 9: 상관 서브쿼리 (EXISTS) — 연락처 레코드가 있는 직원 정보 조회

```
-- EXISTS 상관 서브쿼리: 서브쿼리가 부모 쿼리의 열을 참조
SELECT e.id, e.name
FROM employee e
WHERE EXISTS (
    SELECT ec.id
    FROM employee_connection ec
    WHERE e.id = ec.id
);
```

> ** 📝 결과에 어떤 직원이 포함됩니까?**

```
SELECT *
FROM employee_address
WHERE id NOT IN (SELECT id FROM
employee);
```

> ** 📝 哪些员工地址记录没有对应的员工基本信息？**
>
> 答：_____

练习 9：相关子查询（EXISTS）——查询存在联系方式记录的员工信息

```
-- EXISTS 相关子查询：子查询引用了父查询的列
SELECT e.id, e.name
FROM employee e
WHERE EXISTS (
    SELECT ec.id
    FROM employee_connection ec
    WHERE e.id = ec.id
);
```

> ** 📝 结果有哪些员工？**

>

> 답: _____

제 8 부: CTE 공통 테이블 표현식 (20 분)

학습 목표

- CTE 의 개념과 사용 시나리오 이해

- WITH...AS 문법 습득

- 여러 CTE 를 조합하여 복잡한 조회 수행

> ** 💡 CTE 개념**: 공통 테이블 표현식 (Common Table Expression) 은 이름이 부여된 임시 결과 집합으로, 단일 SQL 문의 실행 범위 내에서 유효합니다. WITH 절을 사용하여 정의하며, SELECT/INSERT 바로 앞에 위치합니다.

연습 1: 단일 CTE 사용 — 캘리포니아에서 사망자 수가 1,000 을

>

> 答：_____

第八部分：CTE 公用表表达式（20 分钟）

学习目标

- 理解 CTE 的概念和使用场景

- 掌握 WITH...AS 语法

- 能使用多个 CTE 组合复杂查询

> ** 💡 CTE 概念**：公用表表达式（Common Table Expression）是一个命名的临时结果集，在单个 SQL 语句的执行范围内有效。使用 WITH 子句定义，紧接在 SELECT/INSERT 之前。

练习 1：使用单个 CTE——获取死亡人数大于 1000 的加州县

초과하는 카운티 조회

```
WITH high_deaths AS (
    SELECT county, cases, deaths
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    AND state = "California"
    AND deaths > 1000
)
SELECT * FROM high_deaths
ORDER BY deaths DESC;
```

> ** 📝 조건에 해당하는 카운티는 몇 개입니까?**
>
> 답: _____

연습 2: CTE 를 사용하여 각 주의 통계 지표 계산 후 필터링

```
WITH state_stats AS (
    SELECT
        state,
        sum(cases)  AS
total_cases,
```

```
WITH high_deaths AS (
    SELECT county, cases, deaths
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    AND state = "California"
    AND deaths > 1000
)
SELECT * FROM high_deaths
ORDER BY deaths DESC;
```

> ** 📝 共有多少个县符合条件？**
>
> 答：_____

练习 2：使用 CTE 计算各州统计指标，再筛选

```
WITH state_stats AS (
    SELECT
        state,
        sum(cases)  AS
total_cases,
```

```
        sum(deaths) AS
total_deaths,
        count(county) AS
county_count
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    GROUP BY state
)
SELECT state, total_cases,
total_deaths
FROM state_stats
WHERE total_deaths > 5000
ORDER BY total_deaths DESC;
```

```
        sum(deaths) AS
total_deaths,
        count(county) AS
county_count
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    GROUP BY state
)
SELECT state, total_cases,
total_deaths
FROM state_stats
WHERE total_deaths > 5000
ORDER BY total_deaths DESC;
```

> ** 📝 결과에 포함된 주는 어디입니까?**
>
> 답: _____

> ** 📝 结果有哪些州？**
>
> 答：_____

연습 3: 여러 CTE 사용 — 고확진 카운티와 고사망 카운티를 각각 계산한 후 결합

练习 3：使用多个 CTE——分别计算高确诊县和高死亡县，然后关联

```
WITH high_cases AS (
    SELECT county, state, cases
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    AND cases > 100000
```

```
WITH high_cases AS (
    SELECT county, state, cases
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    AND cases > 100000
```

```
),
high_deaths AS (
    SELECT county, state, deaths
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    AND deaths > 2000
)
SELECT hc.county, hc.state,
hc.cases, hd.deaths
FROM high_cases hc
INNER JOIN high_deaths hd
ON hc.county = hd.county AND
hc.state = hd.state;
```

```
),
high_deaths AS (
    SELECT county, state, deaths
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-
28"
    AND deaths > 2000
)
SELECT hc.county, hc.state,
hc.cases, hd.deaths
FROM high_cases hc
INNER JOIN high_deaths hd
ON hc.county = hd.county AND
hc.state = hd.state;
```

> ** 📝 고확진과 고사망 조건을 동시에 만족하는 카운티는 어디입니까?**
>
> 답: _____

> ** 📝 同时满足高确诊和高死亡条件的县有哪些？**
>
> 答：_____

연습 4（심화）: CTE 를 사용하여 제 4 부 연습 9 의 SQL 을 다시 작성하세요（평균 연령이 19 세 초과이고 인원이 3 명 초과인 학과 찾기）

练习 4（扩展）：使用 CTE 重写第四部分练习 9 的 SQL（找出平均年龄超过 19 岁且人数大于 3 人的系别）

```
-- WITH...AS 문법으로 다시
작성하세요:
```

```
-- 请用 WITH...AS 语法改写：
```

☑ CTE 와 서브쿼리 비교

특성	CTE（WITH...AS）	서브쿼리（Subquery）
가독성	높음, 논리가 명확	중첩이 많으면 읽기 어려움
재사용 가능 여부	같은 문 내에서 여러 번 참조 가능	재사용 불가
실행 범위	단일 SQL 문 내	단일 서브쿼리 내
적합한 시나리오	복잡한 다단계 조회	단순 필터링

제 9 부: JOIN 연결 연산（45 분）

학습 목표

☑ CTE 与子查询的比较

特性	CTE（WITH...AS）	子查询（Subquery）
可读性	更高，逻辑清晰	嵌套层次多时较难读
是否可复用	同语句内可多次引用	不可复用
执行范围	单个 SQL 语句内	单个子查询
适用场景	复杂多步骤查询	简单过滤

第九部分：JOIN 连接操作（45 分钟）

学习目标

- Hive 가 지원하는 6 가지 JOIN 유형 습득

- 각 JOIN 의 작동 원리와 반환 결과 차이 이해

- 비즈니스 요구에 따라 적절한 JOIN 유형 선택

> ** 💡 JOIN 개념 복습**: JOIN 은 두 개 이상의 테이블에서 열 간의 관계를 기반으로 데이터를 결합하여 조회하는 데 사용됩니다.

>

> **Hive 3.1.2 가 지원하는 6 가지 JOIN**:

> 1. INNER JOIN（내부 조인）

> 2. LEFT JOIN（왼쪽 외부 조인）

> 3. RIGHT JOIN（오른쪽 외부 조인）

> 4. FULL OUTER JOIN（전체 외부 조인）

> 5. LEFT SEMI JOIN（왼쪽 반조인）

> 6. CROSS JOIN（교차 조인/카르테시안 곱）

- 掌握 Hive 支持的 6 种 JOIN 类型

- 理解各种 JOIN 的工作原理和返回结果差异

- 能根据业务需求选择合适的 JOIN 类型

> ** 💡 JOIN 概念回顾**：JOIN 用于根据两个或多个表中列之间的关系，从这些表中联合查询数据。

>

> **Hive 3.1.2 支持的 6 种 JOIN**：

> 1. INNER JOIN（内连接）

> 2. LEFT JOIN（左外连接）

> 3. RIGHT JOIN（右外连接）

> 4. FULL OUTER JOIN（全外连接）

> 5. LEFT SEMI JOIN（左半开连接）

> 6. CROSS JOIN（交叉连接/笛卡尔积）

**세 직원 테이블 데이터 확인 (ID 대응 관계 확인) **

查看三张员工表的数据（确认 ID 的对应关系）

```
SELECT '=== employee ===' AS info;
SELECT * FROM employee;

SELECT '=== employee_address ==='
AS info;
SELECT * FROM employee_address;

SELECT '=== employee_connection
===' AS info;
SELECT * FROM employee_connection;
```

```
SELECT '=== employee ===' AS info;
SELECT * FROM employee;

SELECT '=== employee_address ==='
AS info;
SELECT * FROM employee_address;

SELECT '=== employee_connection
===' AS info;
SELECT * FROM employee_connection;
```

> ** 📝 세 테이블의 ID 교집합 분석**:

>

> - employee 의 id: _____

> - employee_address 의 id: _____

> - employee_connection 의 id: _____

> - 세 테이블 공통 id: _____

9.1 INNER JOIN 내부 조인

> ** 💡 규칙**: 두 테이블에서 **모두

> ** 📝 分析三张表的 ID 交集情况** :

>

> - employee 的 id：_____

> - employee_address 的 id：_____

> - employee_connection 的 id：_____

> - 三表共同拥有的 id：_____

9.1 INNER JOIN 内连接

> ** 💡 规则**：只返回两个表中**都有匹配

매칭**되는 행만 반환합니다.

**的行。

연습 1: 직원 정보와 주소 정보를 동시에 보유한 직원 조회

练习 1：查询同时有员工信息和住址信息的员工

```
-- inner join (inner 는 생략 가능)
SELECT e.id, e.name, e_a.city,
e_a.street
FROM employee e
INNER JOIN employee_address e_a
ON e.id = e_a.id;
```

```
-- inner join（inner 可以省略）
SELECT e.id, e.name, e_a.city,
e_a.street
FROM employee e
INNER JOIN employee_address e_a
ON e.id = e_a.id;
```

> ** 📝 결과는 총 몇 행입니까? 어떤 직원이 포함됩니까?**
>
> 답: _____

> ** 📝 结果共几行？哪些员工出现在结果中？**
>
> 答：_____

연습 2: 세 테이블 INNER JOIN — 주소와 연락처를 모두 보유한 직원 완전 정보 조회

练习 2：三表 INNER JOIN——查询同时有住址和联系方式的员工完整信息

```
SELECT e.id, e.name, e_a.city,
e_conn.phno, e_conn.email
```

```
SELECT e.id, e.name, e_a.city,
e_conn.phno, e_conn.email
```

```
FROM employee e
INNER JOIN employee_address e_a ON e.id = e_a.id
INNER JOIN employee_connection e_conn ON e.id = e_conn.id;
```

```
FROM employee e
INNER JOIN employee_address e_a ON e.id = e_a.id
INNER JOIN employee_connection e_conn ON e.id = e_conn.id;
```

> ** 📝 세 테이블 JOIN 후 결과는 총 몇 행입니까?**
>
> 답: _____

> ** 📝 三表 JOIN 后，结果共几行？**
>
> 答：_____

9.2 LEFT JOIN 왼쪽 외부 조인

9.2 LEFT JOIN 左外连接

> ** 💡 규칙**: 왼쪽 테이블을 기준으로 왼쪽 테이블의 모든 행을 반환하고, 오른쪽 테이블에서 매칭되는 경우 데이터를 표시하고, 매칭되지 않는 경우 NULL 을 표시합니다.

> ** 💡 规则**：以左表为基准，返回左表所有行；右表匹配上的显示数据，匹配不上的显示 NULL。

연습 3: 모든 직원 및 연락처 조회 (연락처가 없는 직원도 표시)**

练习 3：查询所有员工及其联系方式 (没有联系方式的员工也要显示)**

```
SELECT e.id, e.name, e_conn.phno, e_conn.email
```

```
SELECT e.id, e.name, e_conn.phno, e_conn.email
```

```
FROM employee e
LEFT JOIN employee_connection
e_conn
ON e.id = e_conn.id;
```

> ** 📝 결과는 총 몇 행입니까? 어떤 직원의 연락처가 NULL 로 표시됩니까?**

>

> 답: _____

연습 4: LEFT JOIN 실제 활용 — 각 직원의 주소 레코드 유무 통계**

```
SELECT e.id, e.name,
       CASE WHEN e_a.id IS NULL
THEN '주소 정보 없음' ELSE e_a.city
END AS address_status
FROM employee e
LEFT JOIN employee_address e_a ON
e.id = e_a.id;
```

9.3 RIGHT JOIN 오른쪽 외부 조인

```
FROM employee e
LEFT JOIN employee_connection
e_conn
ON e.id = e_conn.id;
```

> ** 📝 结果共几行？哪个员工的联系方式显示为 NULL？**

>

> 答：_____

练习 4：LEFT JOIN 的实际应用——统计每个员工有无住址记录**

```
SELECT e.id, e.name,
       CASE WHEN e_a.id IS NULL
THEN '无住址记录' ELSE e_a.city END
AS address_status
FROM employee e
LEFT JOIN employee_address e_a ON
e.id = e_a.id;
```

9.3 RIGHT JOIN 右外连接

> ** 💡 규칙**: 오른쪽 테이블을 기준으로 오른쪽 테이블의 모든 행을 반환하고, 왼쪽 테이블에서 매칭되지 않는 경우 NULL 을 표시합니다.

> ** 💡 规则**：以右表为基准，返回右表所有行；左表匹配不上的显示 NULL。

연습 5: employee_connection 을 오른쪽 테이블로 하여 모든 연락처 및 해당 직원 정보 조회**

练习 5：以 employee_connection 为右表，查询所有联系方式及对应员工信息**

```
SELECT e.id, e.name, e_conn.phno,
e_conn.email
FROM employee e
RIGHT JOIN employee_connection
e_conn
ON e.id = e_conn.id;
```

```
SELECT e.id, e.name, e_conn.phno,
e_conn.email
FROM employee e
RIGHT JOIN employee_connection
e_conn
ON e.id = e_conn.id;
```

> ** 📝 어떤 연락처 레코드에 해당하는 직원 기본 정보가 없습니까（NULL 표시）?**

>

> 답: _____

> ** 📝 哪条联系方式记录没有对应的员工基本信息（显示 NULL）？**

>

> 答：_____

> ** 📝 사고 문제**: 위의 RIGHT JOIN 조회를 LEFT JOIN 으로 변환하려면（테이블 위치 바꾸기）?

> ** 📝 思考题**：如何用 LEFT JOIN 改写以上 RIGHT JOIN 查询（互换表的位置）？

>

> ```sql

> -- 다시 작성하세요:

> SELECT e.id, e.name, e_conn.phno, e_conn.email

> FROM employee_connection e_conn

> _____ JOIN employee e

> ON _____;

> ```

9.4 FULL OUTER JOIN 전체 외부 조인

> **💡 규칙**: 매칭 여부와 관계없이 왼쪽 테이블과 오른쪽 테이블의 **모든 행**을 반환합니다. 매칭되지 않는 쪽은 NULL 로 표시됩니다.

연습 6: employee 와 employee_address 의 전체 연결 상태 조회**

```
SELECT e.id, e.name, e_a.city, e_a.street
FROM employee e
FULL OUTER JOIN employee_address
```

>

> ```sql

> -- 请改写：

> SELECT e.id, e.name, e_conn.phno, e_conn.email

> FROM employee_connection e_conn

> _____ JOIN employee e

> ON _____;

> ```

9.4 FULL OUTER JOIN 全外连接

> **💡 规则**：返回左表和右表的**所有行**，无论是否有匹配。不匹配的一侧显示 NULL。

练习 6：查询 employee 和 employee_address 的完整关联情况**

```
SELECT e.id, e.name, e_a.city, e_a.street
FROM employee e
FULL OUTER JOIN employee_address
```

```
e_a
ON e.id = e_a.id;
```

```
e_a
ON e.id = e_a.id;
```

> ** 📝 결과는 총 몇 행입니까? 어떤 행에 NULL 값이 나타납니까?**

> ** 📝 结果共几行？哪些行出现了 NULL 值？**

>

> 답: _____

>

> 答：_____

연습 7（사고）: 아래 표를 작성하여 INNER, LEFT, RIGHT, FULL OUTER JOIN 의 결과 행 수 비교

练习 7（思考）：填写下表，对比 INNER、LEFT、RIGHT、FULL OUTER JOIN 的结果行数

| JOIN 유형 | 왼쪽 테이블(employee) 행 수 | 오른쪽 테이블(employee_address) 행 수 | 결과 행 수 |

|---|---|---|---|

| INNER JOIN | 5 | 5 | |

| LEFT JOIN | 5 | 5 | |

| RIGHT JOIN | 5 | 5 | |

| FULL OUTER JOIN | 5 | 5 | |

| JOIN 类型 | 左表(employee)行数 | 右表(employee_address)行数 | 结果行数 |

|---|---|---|---|

| INNER JOIN | 5 | 5 | |

| LEFT JOIN | 5 | 5 | |

| RIGHT JOIN | 5 | 5 | |

| FULL OUTER JOIN | 5 | 5 | |

9.5 LEFT SEMI JOIN 왼쪽 반조인

> ** 💡 규칙**: 오른쪽 테이블에 매칭이 있는 왼쪽 테이블의 행을 반환합니다（EXISTS 서브쿼리와 유사）. **오른쪽 테이블의 어떤 열도 반환하지 않습니다**.

연습 8: 주소 레코드가 있는 직원 정보 조회（LEFT SEMI JOIN 사용）

```
SELECT *
FROM employee e
LEFT SEMI JOIN employee_address
e_addr
ON e.id = e_addr.id;
```

> ** 📝 결과는 총 몇 행입니까? INNER JOIN 결과와 어떻게 다릅니까?**

>

> 답: _____

연습 9（비교）: INNER JOIN 으로

9.5 LEFT SEMI JOIN 左半开连接

> ** 💡 规则**：返回左表中，在右表中有匹配的行（类似 EXISTS 子查询），**不返回右表的任何列**。

练习 8：查询有住址记录的员工信息（使用 LEFT SEMI JOIN）

```
SELECT *
FROM employee e
LEFT SEMI JOIN employee_address
e_addr
ON e.id = e_addr.id;
```

> ** 📝 结果共几行？与 INNER JOIN 的结果有何不同？**

>

> 答：_____

练习 9（对比）：用 INNER JOIN 实现

동일한 결과를 구현하고 비교

相同效果，并对比结果

```
SELECT e.*
FROM employee e
INNER JOIN employee_address e_addr
ON e.id = e_addr.id;
```

```
SELECT e.*
FROM employee e
INNER JOIN employee_address e_addr
ON e.id = e_addr.id;
```

> ** 📝 LEFT SEMI JOIN 과 INNER JOIN 의 핵심 차이점은 무엇입니까?**
>
> 답: _____

> ** 📝 LEFT SEMI JOIN 与 INNER JOIN 的关键区别是什么？**
>
> 答：_____

9.6 CROSS JOIN 교차 조인 (카르테시안 곱)

9.6 CROSS JOIN 交叉连接（笛卡尔积）

> ** ⚠️ 주의**: CROSS JOIN 은 카르테시안 곱 (두 테이블 행 수의 곱) 을 생성하므로, 데이터가 많을 때는 주의해서 사용하세요!

> ** ⚠️ 注意**：CROSS JOIN 会产生笛卡尔积（两表行数的乘积），数据量大时慎用！

****연습 10**: employee 와 employee_address 를 CROSS JOIN**

****练习 10**：对 employee 和 employee_address 进行 CROSS JOIN**

```
SELECT a.id, a.name, b.id AS
addr_id, b.city
FROM employee a
CROSS JOIN employee_address b
LIMIT 10;
```

```
SELECT a.id, a.name, b.id AS
addr_id, b.city
FROM employee a
CROSS JOIN employee_address b
LIMIT 10;
```

> ** 📝 employee(5 행) × employee_address(5 행) = 몇 행입니까?**
>
> 답: _____

> ** 📝 employee(5 行) × employee_address(5 行) = 多少行？**
>
> 答：_____

연습 11: CROSS JOIN 에 WHERE 조건 추가하여 필터링 (INNER JOIN 과 동일)**

练习 11：CROSS JOIN 加 WHERE 条件过滤（等价于 INNER JOIN）**

```
-- 아래 두 조회의 결과가 같습니까?
-- 방식 1: CROSS JOIN + WHERE
SELECT a.id, a.name, b.city
FROM employee a
CROSS JOIN employee_address b
WHERE a.id = b.id;

-- 방식 2: INNER JOIN
SELECT a.id, a.name, b.city
FROM employee a
INNER JOIN employee_address b
ON a.id = b.id;
```

```
-- 下面两个查询结果是否相同？
-- 方式 1：CROSS JOIN + WHERE
SELECT a.id, a.name, b.city
FROM employee a
CROSS JOIN employee_address b
WHERE a.id = b.id;

-- 方式 2：INNER JOIN
SELECT a.id, a.name, b.city
FROM employee a
INNER JOIN employee_address b
ON a.id = b.id;
```

> ** 📝 두 방식의 결과가 같습니까?**:

☑️ 6 가지 JOIN 유형 종합 비교

| JOIN 유형 | 왼쪽 테이블 미매칭 행 반환 | 오른쪽 테이블 미매칭 행 반환 | 오른쪽 테이블 열 반환 | 특징 |

|---|---|---|---|---|

| INNER JOIN | ❌ | ❌ | ✅ | 양쪽 모두 매칭되는 행만 반환 |

| LEFT JOIN | ✅ (NULL) | ❌ | ✅ | 왼쪽 테이블 전체 보존 |

| RIGHT JOIN | ❌ | ✅ (NULL) | ✅ | 오른쪽 테이블 전체 보존 |

| FULL OUTER | ✅ (NULL) | ✅ (NULL) | ✅ | 두 테이블 전체 보존 |

| LEFT SEMI | ❌ | ❌ | ❌ | 왼쪽이 오른쪽에 존재하면 반환 |

| CROSS JOIN | — | — | ✅ | 카르테시안 곱,

> ** 📝 两种方式结果是否相同？** :

☑️ 六种 JOIN 类型综合对比

| JOIN 类型 | 返回左表无匹配行 | 返回右表无匹配行 | 返回右表列 | 特点 |

|---|---|---|---|---|

| INNER JOIN | ❌ | ❌ | ✅ | 只返回双方都匹配的行 |

| LEFT JOIN | ✅ (NULL) | ❌ | ✅ | 左表全部保留 |

| RIGHT JOIN | ❌ | ✅ (NULL) | ✅ | 右表全部保留 |

| FULL OUTER | ✅ (NULL) | ✅ (NULL) | ✅ | 两表全部保留 |

| LEFT SEMI | ❌ | ❌ | ❌ | 左表在右表中存在即返回 |

| CROSS JOIN | — | — | ✅ | 笛卡尔积，慎用

주의해서 사용 |

제 10 부: 종합 도전 과제 (40 분)

학습 목표

- 이 장의 모든 지식 포인트를 종합적으로 활용

- 실제 비즈니스 시나리오 문제 해결

> ** 힌트**: 다음 문제들은 WHERE, GROUP BY, HAVING, JOIN, 서브쿼리, CTE 등 여러 기술을 종합적으로 사용해야 할 수 있습니다. 유연하게 조합하세요.

도전 1: 각 주의 코로나 19 데이터 현황 통계

요구사항: 2021-01-28 기준 각 주의 카운티

|

第十部分：综合挑战练习（40 分钟）

学习目标

- 综合运用本章所有知识点

- 解决真实业务场景问题

> ** 提示**：以下题目可能需要综合使用 WHERE、GROUP BY、HAVING、JOIN、子查询、CTE 等多种技术，请灵活组合。

挑战 1：统计每个州的新冠数据概况

要求：查询 2021-01-28 每个州的县数量、

수, 총 확진 수, 총 사망 수, 평균 확진 수（소수점 2 자리 유지）를 조회하고, 총 확진 수를 기준으로 내림차순 정렬합니다.

总确诊数、总死亡数、平均确诊数（保留 2 位小数），并按总确诊数降序排列。

```
-- SQL 문을 직접 작성하세요:
```

```
-- 请独立编写 SQL 语句：
```

> ** 확진 수가 가장 많은 상위 3 개 주는?**

> ** 确诊数最多的 3 个州是？**

>

> 답: _____

>

> 答：_____

도전 2: 고위험 카운티 찾기（각 주 평균 확진 수보다 확진 수가 많은 카운티）**

挑战 2：找出高风险县（确诊数大于各州平均确诊数的县）**

요구사항: CTE 또는 서브쿼리를 사용하여 각 주 내에서 해당 주의 평균 확진 수를 초과하는 카운티를 찾습니다.

要求：使用 CTE 或子查询，找出每个州内确诊数超过该州平均确诊数的县。

```
-- 힌트: 먼저 CTE 로 주별 평균 확진수 계산 후, JOIN 으로 필터링
```

```
-- 提示：先用 CTE 计算每州平均确诊数，再 JOIN 筛选
```

**도전 3: 학생 테이블 다차원 분석

**挑战 3：学生表多维度分析

① 각 학과의 남녀 학생 수 및 평균 연령 통계 (학과와 성별로 그룹화)

① 统计各系男女生人数及平均年龄（按系别和性别分组）

```
-- SQL 작성:
```

```
-- 请编写 SQL：
```

② 인원이 가장 많은 학과 찾기 (서브쿼리 또는 CTE 사용)

② 找出人数最多的系别（使用子查询或 CTE）

```
-- SQL 작성:
```

```
-- 请编写 SQL：
```

③ 컴퓨터학과(CS) 평균 연령보다 나이가 많은 다른 학과 학생 조회

③ 查询比计算机系(CS)平均年龄大的其他系学生

```
-- SQL 작성:
```

```
-- 请编写 SQL：
```

****도전 4**: 직원 정보 완전성 분석**

****挑战 4**：员工信息完整性分析**

요구사항: 모든 직원을 조회하고, 기본 정보, 주소 도시（주소 없음 시 "미등록"）, 연락 이메일（이메일 없음 시 "미입력"）을 표시합니다.

要求：查询所有员工，显示其基本信息、住址城市（无住址显示"未知"）、联系邮箱（无邮箱显示"未留"）。

```
-- 힌트: FULL OUTER JOIN 또는 여러 LEFT JOIN 사용
-- 힌트: COALESCE() 또는 CASE WHEN 으로 NULL 값 처리
```

```
-- 提示：使用 FULL OUTER JOIN 或多个 LEFT JOIN
-- 提示：使用 COALESCE() 或 CASE WHEN 处理 NULL 值
```

도전 5: NEW YORK 주 카운티 데이터 분석

다음 단계로 분석을 완료하세요:

1. CTE 를 사용하여 New York 주의 모든 카운티 데이터 조회

2. 사망률 계산（deaths/cases × 100）

3. 사망률이 가장 높은 카운티 3 개 찾기

4. ORDER BY 와 LIMIT 을 사용하여 결과 출력

```
WITH ny_data AS (
    -- 1 단계: CTE 내용을 작성하세요

)
-- 2, 3, 4 단계: 여기에 메인 쿼리를 작성하세요
```

挑战 5：NEW YORK 州县级数据分析

使用以下步骤完成分析：

1. 使用 CTE 获取 New York 州的所有县数据

2. 计算死亡率（deaths/cases × 100）

3. 找出死亡率最高的 3 个县

4. 使用 ORDER BY 和 LIMIT 输出结果

```
WITH ny_data AS (
    -- 第 1 步：请填写 CTE 内容

)
-- 第 2、3、4 步：请在此编写主查询
```

🏆 자기 평가 체크리스트

이번 실습을 완료한 후, 다음 지식 포인트에 대해 자가 평가하세요（✅ 완전 습득 / ⚠️ 기본 습득 / ❌ 복습 필요）：

지식 포인트	자기 평가
SELECT 기초 문법（select_expr、정규표현식 열 선택）	
ALL 과 DISTINCT 의 차이	
WHERE 절의 사용법과 제한	
파티션 테이블과 파티션 프루닝	
GROUP BY 의 문법 규칙	
집계 함수（sum/count/avg/max/min）	
HAVING 과 WHERE 의 차이	
LIMIT 의 단일 파라미터와 이중 파라미터 사용법	
SQL 실행 순서	

🏆 自我评估清单

完成本次实践后，请对以下知识点进行自评（✅ 已掌握 / ⚠️ 基本掌握 / ❌ 需复习）：

知识点	自评
SELECT 基础语法（select_expr、正则选列）	
ALL 与 DISTINCT 的区别	
WHERE 子句的用法和限制	
分区表与分区裁剪	
GROUP BY 的语法规则	
聚合函数（sum/count/avg/max/min）	
HAVING 与 WHERE 的区别	
LIMIT 的单参数和双参数用法	
SQL 执行顺序	

| ORDER BY 전역 정렬 | |

| CLUSTER BY 의 원리와 한계 | |

| DISTRIBUTE BY + SORT BY | |

| UNION 과 UNION ALL | |

| FROM 절 서브쿼리 | |

| WHERE 절 서브쿼리（IN/EXISTS） | |

| CTE（WITH...AS）문법 | |

| INNER JOIN | |

| LEFT / RIGHT JOIN | |

| FULL OUTER JOIN | |

| LEFT SEMI JOIN | |

| CROSS JOIN | |

📌 실습 정리

오늘 발생한 주요 문제:

1. _____

2. _____

| ORDER BY 全局排序 | |

| CLUSTER BY 的原理与局限 | |

| DISTRIBUTE BY + SORT BY | |

| UNION 与 UNION ALL | |

| FROM 子句子查询 | |

| WHERE 子句子查询（IN/EXISTS） | |

| CTE（WITH...AS）语法 | |

| INNER JOIN | |

| LEFT / RIGHT JOIN | |

| FULL OUTER JOIN | |

| LEFT SEMI JOIN | |

| CROSS JOIN | |

📌 实践总结

今天遇到的主要问题：

1. _____

2. _____

3. _____

해결 방법:

1. _____

2. _____

3. _____

학습 소감:

_____

_____

《Hive 데이터 웨어하우스 활용 교재》— 프로젝트 6 종합 실습 워크북

3. _____

解决方法：

1. _____

2. _____

3. _____

心得体会：

_____

_____

《Hive 数据仓库应用教程》— 项目六综合实践工作手册