Hive DDL 실습 워크북 (제 4 장)

《Hive 데이터웨어하우스 응용》— 데이터 정의 언어 실습

워크북 목차

번호	실습 내용	권장 시간
준비	실습 환경 확인 및 데이터셋 준비	20 분
실습 4-1	HiveQL 데이터베이스 조작	30 분
실습 4-2	HiveQL 데이터 테이블 조작（내부 테이블/외부 테이블）	60 분
실습 4-3	HiveQL 파티션 테이블 조작	45 분
실습 4-4	HiveQL 버킷 테이블 조작	35 분
실습 4-5	HiveQL 뷰(View) 조작	30 분
실습 4-6	HiveQL 종합 DDL 조작（인덱스 및 Show 구문）	30 분
종합연습	종합 연습문제	30 분
합계		**약 4 시간**

실습 환경 안내

기본 환경

- 운영체제：Ubuntu Linux（VMware 가상머신）
- Hive 클라이언트 도구：beeline（별칭 `hive-bee`）
- Hive 데이터베이스 경로：`/user/hive/warehouse/unit4db`
- 데이터 파일 디렉터리：`/usr/datas/unit4/`

Hive DDL 实战工作手册（第 4 章）

《Hive 数据仓库应用》— 数据定义语言实习

工作手册目录

编号	任务内容	建议用时
准备	实验环境确认与数据集准备	20 分钟
任务 4-1	HiveQL 操作数据库	30 分钟
任务 4-2	HiveQL 操作数据表（内部表/外部表）	60 分钟
任务 4-3	HiveQL 操作分区表	45 分钟
任务 4-4	HiveQL 操作分桶表	35 分钟
任务 4-5	HiveQL 操作视图	30 分钟
任务 4-6	HiveQL 综合 DDL 操作（索引与 Show 语法）	30 分钟
综合练习	综合练习题	30 分钟
合计		**约 4 小时**

实验环境说明

基础环境

- 操作系统：Ubuntu Linux（VMware 虚拟机）
- Hive 客户端工具：beeline（别名 `hive-bee`）
- Hive 数据库路径：`/user/hive/warehouse/unit4db`
- 数据文件目录：`/usr/datas/unit4/`

접속 방법

터미널을 열고 아래 명령어를 입력하여 beeline 을 실행합니다 :

```bash
hive-bee
```

> **안내 : ** `hive-bee`는 시스템에 미리 설정된 beeline 단축 명령어입니다. 실행 후 `beeline>` 프롬프트가 표시되면 정상 접속된 것입니다.

MySQL 정보 (필요한 경우)

- 사용자명 : `hiveuser`
- 비밀번호 : `123456`
- 데이터베이스명 : `unit4db`

데이터셋 파일 경로

모든 실습 데이터 파일은 `/usr/datas/unit4/` 에 저장됩니다.

파일명	설명
`employees.txt`	직원 정보 데이터
`products.txt`	제품 정보 데이터
`sales.txt`	판매 기록 데이터 (파티션 테이블용)
`orders.txt`	주문 데이터 (버킷 테이블용)

🔧 준비 작업 : 데이터셋 생성 (20 분)

실습을 시작하기 전에 Linux 시스템에 데이터 파일을 준비하고, Hive 에 실습 데이터베이스를 생성합니다.

连接方式

打开终端，输入以下命令启动 beeline :

```bash
hive-bee
```

> **说明 : ** `hive-bee` 是系统预置的 beeline 快捷命令，执行后进入 `beeline>` 提示符，表示连接成功。

MySQL 信息 (如需使用)

- 用户名 : `hiveuser`
- 密码 : `123456`
- 数据库名 : `unit4db`

数据集文件路径

所有实习数据文件存放在 : `/usr/datas/unit4/`

文件名	描述
`employees.txt`	员工信息数据
`products.txt`	产品信息数据
`sales.txt`	销售记录数据 (用于分区表)
`orders.txt`	订单数据 (用于分桶表)

🔧 准备工作 : 数据集创建 (20 分钟)

在进行实习之前，需要先在 Linux 系统中准备好数据文件，并在 Hive 中创建实习数据库。

1 단계 : 데이터 디렉터리 생성

Linux 터미널을 열고 아래 명령어를 실행합니다 :

```bash
sudo mkdir -p /usr/datas/unit4
sudo chmod 777 /usr/datas/unit4
```

2 단계 : 데이터 파일 생성

（1）직원 데이터 파일 `employees.txt` 생성

필드 구성 : `직원 ID, 이름, 부서, 급여, 입사일`

```bash
cat > /usr/datas/unit4/employees.txt << 'EOF'
1001,张伟,技术部,15000,2020-03-15
1002,李芳,市场部,12000,2019-07-20
1003,王磊,技术部,18000,2018-05-10
1004,赵敏,人事部,10000,2021-01-08
1005,陈浩,技术部,20000,2017-11-25
1006,刘洋,市场部,13500,2020-09-01
1007,杨静,财务部,11000,2022-02-14
1008,周强,技术部,16500,2019-04-30
1009,吴雪,市场部,12500,2021-06-18
1010,孙明,人事部,10500,2022-08-22
EOF
```

（2）제품 데이터 파일 `products.txt` 생성

필드 구성 : `제품 ID, 제품명, 카테고리, 가격, 재고수량`

```bash
cat > /usr/datas/unit4/products.txt << 'EOF'
P001,笔记本电脑,电子产品,6999.00,150
P002,无线鼠标,电子产品,99.00,500
P003,机械键盘,电子产品,299.00,320
P004,办公椅,办公家具,1299.00,80
P005,会议桌,办公家具,3500.00,20
P006,文件柜,办公家具,899.00,60
```

步骤 1：创建数据目录

打开 Linux 终端，执行以下命令创建数据目录：

```bash
sudo mkdir -p /usr/datas/unit4
sudo chmod 777 /usr/datas/unit4
```

步骤 2：创建数据文件

（1）创建员工数据文件 `employees.txt`

字段说明：`员工 ID, 姓名, 部门, 薪资, 入职日期`

```bash
cat > /usr/datas/unit4/employees.txt << 'EOF'
1001,张伟,技术部,15000,2020-03-15
1002,李芳,市场部,12000,2019-07-20
1003,王磊,技术部,18000,2018-05-10
1004,赵敏,人事部,10000,2021-01-08
1005,陈浩,技术部,20000,2017-11-25
1006,刘洋,市场部,13500,2020-09-01
1007,杨静,财务部,11000,2022-02-14
1008,周强,技术部,16500,2019-04-30
1009,吴雪,市场部,12500,2021-06-18
1010,孙明,人事部,10500,2022-08-22
EOF
```

（2）创建产品数据文件 `products.txt`

字段说明：`产品 ID, 产品名称, 类别, 价格, 库存数量`

```bash
cat > /usr/datas/unit4/products.txt << 'EOF'
P001,笔记本电脑,电子产品,6999.00,150
P002,无线鼠标,电子产品,99.00,500
P003,机械键盘,电子产品,299.00,320
P004,办公椅,办公家具,1299.00,80
P005,会议桌,办公家具,3500.00,20
P006,文件柜,办公家具,899.00,60
```

```
P007,打印纸,办公耗材,45.00,1000
P008,中性笔,办公耗材,2.50,5000
P009,订书机,办公耗材,35.00,200
P010,投影仪,电子产品,4500.00,30
EOF
```

（3）판매 데이터 파일 `sales.txt` 생성（파티션 테이블용）

필드 구성：`판매 ID, 직원 ID, 제품 ID, 판매금액, 판매일`

```bash
cat > /usr/datas/unit4/sales.txt << 'EOF'
S001,1001,P001,6999.00,2023-01-15
S002,1002,P007,45.00,2023-01-20
S003,1003,P002,99.00,2023-02-05
S004,1001,P010,4500.00,2023-02-18
S005,1005,P003,299.00,2023-03-10
S006,1006,P004,1299.00,2023-03-22
S007,1002,P008,2.50,2023-04-08
S008,1004,P005,3500.00,2023-04-15
S009,1003,P006,899.00,2023-05-02
S010,1007,P001,6999.00,2023-05-20
S011,1001,P002,99.00,2023-06-11
S012,1008,P009,35.00,2023-06-25
EOF
```

（4）주문 데이터 파일 `orders.txt` 생성（버킷 테이블용）

필드 구성：`주문 ID, 고객명, 성(省份), 제품카테고리, 주문금액`

```bash
cat > /usr/datas/unit4/orders.txt << 'EOF'
O001,北京科技有限公司,北京,电子产品,25000.00
O002,上海贸易公司,上海,办公家具,18000.00
O003,广州制造企业,广东,电子产品,32000.00
O004,深圳创新公司,广东,办公耗材,5000.00
O005,北京教育机构,北京,电子产品,45000.00
O006,上海金融公司,上海,办公家具,22000.00
O007,广州零售连锁,广东,办公耗材,8000.00
```


```
P007,打印纸,办公耗材,45.00,1000
P008,中性笔,办公耗材,2.50,5000
P009,订书机,办公耗材,35.00,200
P010,投影仪,电子产品,4500.00,30
EOF
```


（3）创建销售数据文件 `sales.txt`（用于分区表）

字段说明：`销售 ID, 员工 ID, 产品 ID, 销售金额, 销售日期`


```bash
cat > /usr/datas/unit4/sales.txt << 'EOF'
S001,1001,P001,6999.00,2023-01-15
S002,1002,P007,45.00,2023-01-20
S003,1003,P002,99.00,2023-02-05
S004,1001,P010,4500.00,2023-02-18
S005,1005,P003,299.00,2023-03-10
S006,1006,P004,1299.00,2023-03-22
S007,1002,P008,2.50,2023-04-08
S008,1004,P005,3500.00,2023-04-15
S009,1003,P006,899.00,2023-05-02
S010,1007,P001,6999.00,2023-05-20
S011,1001,P002,99.00,2023-06-11
S012,1008,P009,35.00,2023-06-25
EOF
```


（4）创建订单数据文件 `orders.txt`（用于分桶表）

字段说明：`订单 ID, 客户名称, 省份, 产品类别, 订单金额`


```bash
cat > /usr/datas/unit4/orders.txt << 'EOF'
O001,北京科技有限公司,北京,电子产品,25000.00
O002,上海贸易公司,上海,办公家具,18000.00
O003,广州制造企业,广东,电子产品,32000.00
O004,深圳创新公司,广东,办公耗材,5000.00
O005,北京教育机构,北京,电子产品,45000.00
O006,上海金融公司,上海,办公家具,22000.00
O007,广州零售连锁,广东,办公耗材,8000.00
```

```
0008,北京医疗机构,北京,电子产品,38000.00
0009,杭州互联网公司,浙江,电子产品,55000.00
0010,南京工程公司,江苏,办公家具,12000.00
0011,武汉大学,湖北,电子产品,28000.00
0012,成都科技园区,四川,电子产品,42000.00
0013,杭州电商平台,浙江,办公耗材,6500.00
0014,苏州制造业,江苏,办公家具,15000.00
0015,武汉研究院,湖北,电子产品,33000.00
EOF
```

3 단계 : 파일 생성 확인

```bash
ls -la /usr/datas/unit4/
cat /usr/datas/unit4/employees.txt
```

> ✅ **체크포인트 : ** 4 개의 데이터 파일이 모두 생성되고 내용이 정확한지 확인한 후 이후 실습을 진행합니다.

4 단계 : Hive 실습 데이터베이스 생성

beeline 을 실행합니다 :

```bash
hive-bee
```

beeline 에서 아래 SQL 을 실행합니다 :

```sql
-- 실습 데이터베이스 생성
CREATE DATABASE IF NOT EXISTS unit4db
COMMENT '제 4 장 DDL 실습 데이터베이스'
LOCATION '/user/hive/warehouse/unit4db';

-- 데이터베이스 생성 확인
SHOW DATABASES;

-- unit4db 로 전환
USE unit4db;
```

```
0008,北京医疗机构,北京,电子产品,38000.00
0009,杭州互联网公司,浙江,电子产品,55000.00
0010,南京工程公司,江苏,办公家具,12000.00
0011,武汉大学,湖北,电子产品,28000.00
0012,成都科技园区,四川,电子产品,42000.00
0013,杭州电商平台,浙江,办公耗材,6500.00
0014,苏州制造业,江苏,办公家具,15000.00
0015,武汉研究院,湖北,电子产品,33000.00
EOF
```

步骤 3：验证文件创建是否成功

```bash
ls -la /usr/datas/unit4/
cat /usr/datas/unit4/employees.txt
```

> ✅ **检查点：** 确认 4 个数据文件均已创建，且内容正确后，再进行后续实习。

步骤 4：创建 Hive 实习数据库

启动 beeline 并创建数据库：

```bash
hive-bee
```

在 beeline 中执行：

```sql
-- 创建实习数据库
CREATE DATABASE IF NOT EXISTS unit4db
COMMENT '第 4 章 DDL 实习数据库'
LOCATION '/user/hive/warehouse/unit4db';

-- 确认数据库创建成功
SHOW DATABASES;

-- 切换到 unit4db 数据库
USE unit4db;
```

실습 4-1 : HiveQL 데이터베이스 조작 (30 분)

학습 목표

- Hive 데이터베이스의 생성, 조회, 확인, 전환, 수정, 삭제 조작 습득
- CREATE DATABASE, SHOW DATABASES, DESC DATABASE, USE, ALTER DATABASE, DROP DATABASE 구문 숙지

실습 4-1-1 : 데이터베이스 생성

**실습 내용 : ** Hive 에서 본 장 실습용 데이터베이스를 생성하고 HDFS 저장 경로와 설명 정보를 지정합니다.

**조작 순서 : **

```sql
-- 1. 기본 데이터베이스 생성
CREATE DATABASE IF NOT EXISTS unit4db
COMMENT '제 4 장 HiveQL DDL 전용 실습 데이터베이스'
LOCATION '/user/hive/warehouse/unit4db';
```

```sql
-- 2. 사용자 정의 속성이 있는 데이터베이스 생성 (연습)
CREATE DATABASE IF NOT EXISTS unit4db_test
COMMENT '테스트용 데이터베이스'
WITH DBPROPERTIES (
  'creator'     = 'student',
  'create_date' = '2024-01-01',
  'version'     = '1.0'
);
```

任务 4-1 : HiveQL 操作数据库 (30 分钟)

学习目标

- 掌握 Hive 数据库的创建、查询、查看、切换、修改、删除操作
- 熟悉 CREATE DATABASE、SHOW DATABASES、DESC DATABASE、USE、ALTER DATABASE、DROP DATABASE 语法

实习 4-1-1 : 创建数据库

**任务描述 : ** 在 Hive 中创建一个用于本章实习的数据库，并指定 HDFS 存储路径和描述信息。

**操作步骤 : **

```sql
-- 1. 创建基本数据库
CREATE DATABASE IF NOT EXISTS unit4db
COMMENT '第 4 章 HiveQL DDL 实习专用数据库'
LOCATION '/user/hive/warehouse/unit4db';
```

```sql
-- 2. 创建带有自定义属性的数据库 (练习)
CREATE DATABASE IF NOT EXISTS unit4db_test
COMMENT '测试用数据库'
WITH DBPROPERTIES (
  'creator' = 'student',
  'create_date' = '2024-01-01',
  'version' = '1.0'
);
```

> 📝 **문법 핵심 :**

> - `IF NOT EXISTS` : 데이터베이스가 이미 존재할 경우 오류 방지

> - `COMMENT` : 데이터베이스 설명 추가

> - `LOCATION` : HDFS 저장 경로 지정

> - `WITH DBPROPERTIES` : 사용자 정의 속성 (키-값 쌍) 설정

실습 4-1-2 : 데이터베이스 조회

실습 내용 : 현재 Hive 에 있는 모든 데이터베이스를 확인하고, 와일드카드 조회로 특정 데이터베이스를 필터링합니다.

```sql
-- 1. 모든 데이터베이스 조회
SHOW DATABASES;
```

```sql
-- 2. 와일드카드 조회 : "unit"으로 시작하는 데이터베이스 조회
SHOW DATABASES LIKE 'unit*';
```

```sql
-- 3. 와일드카드 조회 : "4"를 포함하는 데이터베이스 조회
SHOW DATABASES LIKE '*4*';
```

> 📝 **문법 핵심 :**

> - `LIKE` 절은 와일드카드 `*` (임의 문자 일치) 와 `|` (논리 OR) 를 지원합니다.

실습 4-1-3 : 데이터베이스 상세 정보 확인

> 📝 **语法要点 :**

> - `IF NOT EXISTS` : 防止数据库已存在时报错

> - `COMMENT` : 为数据库添加描述

> - `LOCATION` : 指定 HDFS 上的存储路径

> - `WITH DBPROPERTIES` : 设置自定义属性 (键值对)

实习 4-1-2 : 查询数据库

任务描述 : 查看当前 Hive 中所有数据库, 并使用模糊查询筛选特定数据库。

```sql
-- 1. 查看所有数据库
SHOW DATABASES;
```

```sql
-- 2. 使用模糊查询 : 查询以 "unit" 开头的数据库

SHOW DATABASES LIKE 'unit*';
```

```sql
-- 3. 使用模糊查询 : 查询包含 "4" 的数据库
SHOW DATABASES LIKE '*4*';
```

> 📝 **语法要点 :**

> - `LIKE` 子句支持通配符 `*` (匹配任意字符) 和 `|` (逻辑或)

实习 4-1-3 : 查看数据库详细信息

**실습 내용 : ** unit4db 데이터베이스의 상세 정보 (설명, 저장 경로, 속성) 를 확인합니다.

```sql
-- 1. 데이터베이스 기본 정보 확인
DESC DATABASE unit4db;
```

```sql
-- 2. 데이터베이스 확장 정보 확인 (속성 포함)
DESC DATABASE EXTENDED unit4db;
```

```sql
-- 3. 속성이 있는 테스트 데이터베이스 상세 정보 확인
DESC DATABASE EXTENDED unit4db_test;
```

> 💡 **관찰 : ** `DESC DATABASE`와 `DESC DATABASE EXTENDED`의 출력 결과를 비교해 보세요. `EXTENDED` 키워드를 사용하면 `WITH DBPROPERTIES`로 설정한 추가 속성이 표시됩니다.

실습 4-1-4 : 데이터베이스 전환

```sql
-- unit4db 로 전환
USE unit4db;

-- 현재 데이터베이스 확인
SELECT current_database();
```

실습 4-1-5 : 데이터베이스 수정

**실습 내용 : ** unit4db_test 데이터베이스의 속성

任务描述： 查看 unit4db 数据库的详细信息，包括描述、存储路径和属性。

```sql
-- 1. 查看数据库基本信息
DESC DATABASE unit4db;
```

```sql
-- 2. 查看数据库扩展信息（包含属性）
DESC DATABASE EXTENDED unit4db;
```

```sql
-- 3. 查看带属性的测试数据库详细信息
DESC DATABASE EXTENDED unit4db_test;
```

> 💡 **观察：** 对比 `DESC DATABASE` 和 `DESC DATABASE EXTENDED` 的输出结果，注意 EXTENDED 关键字会显示通过 `WITH DBPROPERTIES` 设置的额外属性。

实习 4-1-4：切换数据库

```sql
-- 切换到 unit4db 数据库
USE unit4db;

-- 验证当前数据库
SELECT current_database();
```

实习 4-1-5：修改数据库

任务描述： 修改 unit4db_test 数据库的属性信息。

정보를 수정합니다.

```sql
-- 데이터베이스 속성 수정
ALTER DATABASE unit4db_test
SET DBPROPERTIES (
  'last_modified' = '2024-06-01',
  'status'        = 'active'
);

-- 수정 결과 확인
DESC DATABASE EXTENDED unit4db_test;
```

실습 4-1-6 : 데이터베이스 삭제

```sql
-- 테스트 데이터베이스 삭제 (주의 : 빈 데이터베이스만 삭제 가능)
DROP DATABASE IF EXISTS unit4db_test;

-- 삭제 확인
SHOW DATABASES;
```

> ⚠️ **주의 : ** 테이블이 있는 데이터베이스를 삭제하려면 `CASCADE` 옵션을 사용합니다 :
> ```sql
> DROP DATABASE IF EXISTS unit4db_test CASCADE;
> ```

연습문제 4-1

**연습 1 : ** `practice_db`라는 데이터베이스를 생성하세요. 조건 :

- 주석 추가 : "연습용 데이터베이스"
- HDFS 경로 지정 : `/user/hive/warehouse/practice_db`

```sql
-- 修改数据库属性
ALTER DATABASE unit4db_test
SET DBPROPERTIES (
  'last_modified' = '2024-06-01',
  'status' = 'active'
);

-- 验证修改结果
DESC DATABASE EXTENDED unit4db_test;
```

实习 4-1-6 : 删除数据库

```sql
-- 删除测试数据库（注意：只能删除空数据库）

DROP DATABASE IF EXISTS unit4db_test;

-- 确认删除成功
SHOW DATABASES;
```

> ⚠️ **注意 : ** 如果数据库中存在表，需要使用 `CASCADE` 选项 :
> ```sql
> DROP DATABASE IF EXISTS unit4db_test CASCADE;
> ```

练习题 4-1

**练习 1 : ** 创建一个名为 `practice_db` 的数据库，要求 :

- 添加注释 : "练习用数据库"
- 指定 HDFS 路径为 : `/user/hive/warehouse/practice_db`

- 속성 추가 : `author`=`학생이름`, `subject`=`빅데이터기술`

**연습 2 : ** LIKE 절을 사용하여 알파벳 "p"로 시작하는 모든 데이터베이스를 조회하세요.

**연습 3 : ** `practice_db`의 상세 정보를 확인하여 속성이 올바르게 설정되었는지 검증하세요.

**연습 4 : ** `practice_db`에 새 속성 `semester`=`2024-spring`을 추가하고 수정 결과를 확인하세요.

**연습 5 : ** `practice_db` 데이터베이스를 삭제하세요.

실습 4-2 : HiveQL 데이터 테이블 조작 (60 분)

학습 목표

- 내부 테이블 (Managed Table) 과 외부 테이블 (External Table) 생성 방법 습득
- 테이블 조회, 수정, 삭제 조작 습득
- 내부 테이블과 외부 테이블의 차이점 이해

실습 4-2-1 : 내부 테이블 생성 (Managed Table)

**실습 내용 : ** unit4db 데이터베이스에 직원 정보 내부 테이블 `employees`를 생성합니다.

- 添加属性：`author`=`学生姓名`，`subject`=`大数据技术`

练习 2： 使用 LIKE 子句，查询所有以字母 "p" 开头的数据库。

练习 3： 查看 `practice_db` 的详细信息，确认属性是否正确设置。

练习 4： 为 `practice_db` 增加新属性 `semester`=`2024-spring`，并验证修改结果。

练习 5： 删除 `practice_db` 数据库。

任务 4-2：HiveQL 操作数据表（60 分钟）

学习目标

- 掌握内部表（Managed Table）和外部表（External Table）的创建
- 掌握数据表的查看、修改、删除操作
- 理解内部表和外部表的区别

实习 4-2-1：创建内部表（Managed Table）

任务描述： 在 unit4db 数据库中创建员工信息内部表 `employees`。

**조작 전 현재 데이터베이스 확인 : **

```sql
USE unit4db;
SELECT current_database();
```

**내부 테이블 생성 : **

```sql
CREATE TABLE IF NOT EXISTS unit4db.employees (
  emp_id     INT    COMMENT '직원 ID',
  emp_name   STRING COMMENT '직원 이름',
  dept       STRING COMMENT '소속 부서',
  salary     DOUBLE COMMENT '급여',
  hire_date STRING COMMENT '입사일'
)
COMMENT '직원 정보 내부 테이블'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
TBLPROPERTIES ('creator'='student',
'table_type'='managed');
```

> 📝 **문법 핵심 : **

> - `ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' : 필드 구분자를 쉼표로 지정

> - `LINES TERMINATED BY '\n' : 행 구분자를 줄바꿈으로 지정

> - `STORED AS TEXTFILE` : 텍스트 형식으로 저장

> - `TBLPROPERTIES` : 테이블 속성 설정

실습 4-2-2 : 외부 테이블 생성 (External Table)

실습 내용 : ** **unit4db 데이터베이스에 제품 정보 외부 테이블 `products_ext`를 생성하고 외부 저장 경로를 지정합니다.

**操作前确认当前数据库 : **

```sql
USE unit4db;
SELECT current_database();
```

**创建内部表 : **

```sql
CREATE TABLE IF NOT EXISTS unit4db.employees (
  emp_id     INT          COMMENT '员工 ID',
  emp_name   STRING       COMMENT '员工姓名',
  dept       STRING       COMMENT '所属部门',
  salary     DOUBLE       COMMENT '薪资',
  hire_date  STRING       COMMENT '入职日期'
)
COMMENT '员工信息内部表'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
TBLPROPERTIES ('creator'='student',
'table_type'='managed');
```

> 📝 **语法要点 : **

> - `ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' : 指定字段分隔符为逗号

> - `LINES TERMINATED BY '\n' : 指定行分隔符为换行符

> - `STORED AS TEXTFILE` : 以文本格式存储

> - `TBLPROPERTIES` : 设置表属性

实习 4-2-2 : 创建外部表 (External Table)

任务描述 : ** 在 **unit4db 数据库中创建产品信息外部表 `products_ext`，并指定外部存储路径。

```sql
CREATE EXTERNAL TABLE IF NOT EXISTS
unit4db.products_ext (
  prod_id    STRING COMMENT '제품 ID',
  prod_name STRING COMMENT '제품명',
  category   STRING COMMENT '제품 카테고리',
  price      DOUBLE COMMENT '가격',
  stock      INT    COMMENT '재고 수량'
)
COMMENT '제품 정보 외부 테이블'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/hive/warehouse/unit4db/products_ext'
TBLPROPERTIES ('creator'='student',
'table_type'='external');
```

> 📝 **내부 테이블 vs 외부 테이블 비교：**

> | 특성 | 내부 테이블 | 외부 테이블 |

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

> | 키워드 | 없음（기본값） | `EXTERNAL` |

> | 테이블 삭제 시 | 메타데이터 + 데이터 파일 모두 삭제 | 메타데이터만 삭제, 데이터 파일 유지 |

> | 적합한 상황 | Hive 가 단독으로 데이터 관리 | 여러 시스템이 데이터 공유 |

실습 4-2-3：테이블에 데이터 로드

실습 내용： 로컬 파일의 데이터를 Hive 테이블에 로드합니다.

```sql
-- 로컬 employees.txt 데이터를 내부 테이블
employees 에 로드
LOAD DATA LOCAL INPATH
'/usr/datas/unit4/employees.txt'
INTO TABLE unit4db.employees;
```

```sql
CREATE EXTERNAL TABLE IF NOT EXISTS
unit4db.products_ext (
  prod_id    STRING      COMMENT '产品 ID',
  prod_name  STRING      COMMENT '产品名称',
  category   STRING      COMMENT '产品类别',
  price      DOUBLE      COMMENT '价格',
  stock      INT         COMMENT '库存数量'
)
COMMENT '产品信息外部表'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/hive/warehouse/unit4db/products_ext'
TBLPROPERTIES ('creator'='student',
'table_type'='external');
```

> 📝 **内外部表对比：**

> | 特性 | 内部表 | 外部表 |

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

> | 关键字 | 无（默认） | `EXTERNAL` |

> | 删除表时 | 元数据+数据文件均删除 | 只删除元数据, 数据文件保留 |

> | 适用场景 | Hive 独占管理数据 | 多系统共享数据 |

实习 4-2-3：向表中加载数据

任务描述： 将本地文件中的数据加载到 Hive 表中。

```sql
-- 将本地 employees.txt 数据加载到内部表 employees

LOAD DATA LOCAL INPATH
'/usr/datas/unit4/employees.txt'
INTO TABLE unit4db.employees;
```

```sql
-- 로컬 products.txt 데이터를 외부 테이블 products_ext 에 로드
LOAD DATA LOCAL INPATH '/usr/datas/unit4/products.txt'
INTO TABLE unit4db.products_ext;
```

```sql
-- 데이터 로드 확인
SELECT * FROM unit4db.employees;
SELECT COUNT(*) FROM unit4db.employees;
```

> ⚠️ **핵심 주의 : ** `LOCAL` 키워드는 로컬 파일 시스템에서 로드함을 의미합니다. `LOCAL` 없이 사용하면 HDFS 에서 로드합니다.

실습 4-2-4 : 테이블 정보 조회

```sql
-- 1. unit4db 의 모든 테이블 조회
USE unit4db;
SHOW TABLES;
```

```sql
-- 2. 와일드카드 조회 : "e"로 시작하는 테이블 조회
SHOW TABLES LIKE 'e*';
```

```sql
-- 3. 내부 테이블 employees 의 기본 구조 정보 조회
DESC employees;
```

```sql
-- 4. 외부 테이블 products_ext 의 상세 구조 정보 조회 (저장 형식, 파일 경로 등 포함)
DESC FORMATTED products_ext;
```

> 💡 **관찰 포인트 : ** `DESC FORMATTED` 실행 후

```sql
-- 将本地 products.txt 数据加载到外部表 products_ext

LOAD DATA LOCAL INPATH '/usr/datas/unit4/products.txt'
INTO TABLE unit4db.products_ext;
```

```sql
-- 验证数据加载成功
SELECT * FROM unit4db.employees;
SELECT COUNT(*) FROM unit4db.employees;
```

> ⚠️ **关键注意 : ** 使用 `LOCAL` 关键字表示从本地文件系统加载，不加 `LOCAL` 则从 HDFS 加载。

实习 4-2-4 : 查看数据表信息

```sql
-- 1. 查看 unit4db 中所有数据表
USE unit4db;
SHOW TABLES;
```

```sql
-- 2. 模糊查询 : 查看以 "e" 开头的数据表
SHOW TABLES LIKE 'e*';
```

```sql
-- 3. 查看内部表 employees 的基本结构信息
DESC employees;
```

```sql
-- 4. 查看外部表 products_ext 的详细结构信息（含存储格式、文件路径等）
DESC FORMATTED products_ext;
```

> 💡 **观察重点 : ** 执行 `DESC FORMATTED` 后，注意

다음 필드를 확인하세요 :

> - `Table Type` : 내부 테이블 (MANAGED_TABLE) 또는 외부 테이블 (EXTERNAL_TABLE) 판별

> - `Location` : HDFS 의 테이블 데이터 저장 경로

실습 4-2-5 : 테이블 구조 수정

(1) 연습용 테이블 생성

```sql
-- 구조 수정 연습용 테이블 생성
CREATE TABLE IF NOT EXISTS unit4db.alter_test (
  id     STRING COMMENT '번호',
  gender STRING COMMENT '성별',
  name   STRING COMMENT '이름'
);
```

(2) 테이블 이름 변경

```sql
-- alter_test 를 alter_test_new 로 이름 변경
ALTER TABLE alter_test RENAME TO alter_test_new;

-- 결과 확인
SHOW TABLES;
```

(3) 컬럼 정의 수정 (이름 변경, 위치 이동, 주석 수정)

```sql
-- gender 컬럼을 sex 로 이름 변경, name 컬럼 뒤로 이동, 주석 업데이트
ALTER TABLE alter_test_new
CHANGE gender sex STRING
COMMENT '성별 필드'
AFTER name;

-- 수정 결과 확인
DESC alter_test_new;
```

查看以下字段：

> - `Table Type` : 判断是内部表 (MANAGED_TABLE) 还是外部表 (EXTERNAL_TABLE)

> - `Location` : 表数据在 HDFS 上的存储路径

实习 4-2-5 : 修改数据表结构

(1) 创建练习用表

```sql
-- 创建结构修改练习表
CREATE TABLE IF NOT EXISTS unit4db.alter_test (
  id     STRING  COMMENT '编号',
  gender STRING  COMMENT '性别',
  name   STRING  COMMENT '姓名'
);
```

(2) 重命名表

```sql
-- 将 alter_test 重命名为 alter_test_new
ALTER TABLE alter_test RENAME TO alter_test_new;

-- 验证重命名结果
SHOW TABLES;
```

(3) 修改列定义 (重命名列、移动位置、修改注释)

```sql
-- 将 gender 列改名为 sex, 并移动到 name 列之后, 同时更新注释
ALTER TABLE alter_test_new
CHANGE gender sex STRING
COMMENT '性别字段'
AFTER name;

-- 查看修改结果
DESC alter_test_new;
```

（4）컬럼 데이터 타입 수정

```sql
-- sex 컬럼의 데이터 타입을 STRING 에서 VARCHAR(10)으로 수정
ALTER TABLE alter_test_new
CHANGE sex sex VARCHAR(10)
COMMENT '성별 필드';

-- 수정 결과 확인
DESC alter_test_new;
```

（5）새 컬럼 추가

```sql
-- 테이블 끝에 age 와 email 컬럼 추가
ALTER TABLE alter_test_new
ADD COLUMNS (
  age    INT    COMMENT '나이',
  email STRING COMMENT '이메일'
);

-- 수정 결과 확인
DESC alter_test_new;
```

（6）모든 컬럼 교체（테이블 구조 재설정）

```sql
-- 테이블의 모든 컬럼을 교체（주의해서 사용！）
ALTER TABLE alter_test_new
REPLACE COLUMNS (
  user_id   INT    COMMENT '사용자 ID',
  user_name STRING COMMENT '사용자명',
  password  STRING COMMENT '비밀번호'
);

-- 수정 결과 확인
DESC alter_test_new;
```

（7）테이블 속성 수정

```sql
ALTER TABLE alter_test_new
SET TBLPROPERTIES ('last_modified' = '2024-06-01',
'status' = 'test');

-- 테이블 상세 정보 확인
```

（4）修改列的数据类型

```sql
-- 将 sex 列的数据类型由 STRING 改为 VARCHAR(10)

ALTER TABLE alter_test_new
CHANGE sex sex VARCHAR(10)
COMMENT '性别字段';

-- 查看修改结果
DESC alter_test_new;
```

（5）添加新列

```sql
-- 在表末尾添加 age 和 email 两列
ALTER TABLE alter_test_new
ADD COLUMNS (
  age    INT    COMMENT '年龄',
  email STRING COMMENT '电子邮件'
);

-- 查看修改结果
DESC alter_test_new;
```

（6）替换所有列（重置表结构）

```sql
-- 替换表中所有列（谨慎使用！）
ALTER TABLE alter_test_new
REPLACE COLUMNS (
  user_id   INT    COMMENT '用户 ID',
  user_name STRING COMMENT '用户名',
  password  STRING COMMENT '密码'
);

-- 查看修改结果
DESC alter_test_new;
```

（7）修改表属性

```sql
ALTER TABLE alter_test_new
SET TBLPROPERTIES ('last_modified' = '2024-06-01',
'status' = 'test');

-- 查看表详细信息
```

```
DESC FORMATTED alter_test_new;
```

실습 4-2-6 : 테이블 삭제

(1) 연습용 테이블 삭제

```sql
-- alter_test_new 테이블 삭제 (PURGE: 휴지통 거치지 않고 즉시 삭제)
DROP TABLE IF EXISTS alter_test_new PURGE;

-- 삭제 확인
SHOW TABLES;
```

> 💡 **생각해보기 : ** 내부 테이블 `employees`와 외부 테이블 `products_ext`를 삭제하면 HDFS 의 데이터 파일은 어떻게 될까요 ?

**(2) 내부/외부 테이블 삭제 차이 확인 (관찰 실험) **

```sql
-- 내부 테이블 데이터 파일 경로 확인 (삭제 전에 경로 기록)
DESC FORMATTED employees;

-- 외부 테이블 데이터 파일 경로 확인 (삭제 전에 경로 기록)
DESC FORMATTED products_ext;
```

```bash
# Linux 터미널에서 HDFS 파일 확인 (테이블 삭제 후 다시 확인하여 비교)
hdfs dfs -ls /user/hive/warehouse/unit4db/
```

연습문제 4-2


```
DESC FORMATTED alter_test_new;
```

实习 4-2-6 : 删除数据表

(1) 删除练习表

```sql
-- 删除 alter_test_new 表 (PURGE 表示不经过回收站, 直接删除)
DROP TABLE IF EXISTS alter_test_new PURGE;

-- 验证删除成功
SHOW TABLES;
```

> 💡 **思考 : ** 删除内部表 `employees` 和外部表 `products_ext` 后, HDFS 上的数据文件会发生什么变化 ?

**(2) 验证内外部表删除的区别 (观察实验) **

```sql
-- 查看内部表数据文件路径 (删除前记录路径)
DESC FORMATTED employees;

-- 查看外部表数据文件路径 (删除前记录路径)
DESC FORMATTED products_ext;
```

```bash
# 在 Linux 终端检查 HDFS 上的文件 (删除表后再检查对比)
hdfs dfs -ls /user/hive/warehouse/unit4db/
```

练习题 4-2

**연습 1 : ** 내부 테이블 `dept_info`를 생성하세요. 필드 :

- `dept_id` : INT, 부서 번호
- `dept_name` : STRING, 부서명
- `manager` : STRING, 부서장
- `location` : STRING, 사무실 위치
- 필드 구분자를 탭 (`\t`) 으로 설정

**练习 1 : ** 创建内部表 `dept_info`, 字段如下 :

- `dept_id` : INT, 部门编号
- `dept_name` : STRING, 部门名称
- `manager` : STRING, 部门经理
- `location` : STRING, 办公地点
- 字段分隔符设置为制表符 `\t`

**연습 2 : ** 외부 테이블 `salary_grade_ext`를 생성하세요. 필드 :

- `grade_id` : INT, 급여 등급
- `min_salary` : DOUBLE, 최저 급여
- `max_salary` : DOUBLE, 최고 급여
- `grade_name` : STRING, 등급명
- LOCATION 지정 : `/user/hive/warehouse/unit4db/salary_grade`

**练习 2 : ** 创建外部表 `salary_grade_ext`, 字段如下 :

- `grade_id` : INT, 薪资等级
- `min_salary` : DOUBLE, 最低薪资
- `max_salary` : DOUBLE, 最高薪资
- `grade_name` : STRING, 等级名称
- 指定 LOCATION 为 `/user/hive/warehouse/unit4db/salary_grade`

**연습 3 : ** `dept_info`에 두 개의 새 컬럼을 추가하세요 : `phone` (STRING, 연락처) 과 `email` (STRING, 부서 이메일). DESC 명령으로 확인하세요.

**练习 3 : ** 向 `dept_info` 添加两个新列 : `phone` (STRING, 联系电话) 和 `email` (STRING, 部门邮箱), 然后用 DESC 命令验证。

**연습 4 : ** `dept_info` 테이블의 `manager` 컬럼을 `dept_manager`로 이름 변경하고, `dept_name` 컬럼 뒤로 이동하세요.

**练习 4 : ** 将 `dept_info` 表的 `manager` 列重命名为 `dept_manager`, 并将其移动到 `dept_name` 列之后。

**연습 5 : ** `DESC FORMATTED` 명령으로 `dept_info` (내부 테이블) 와 `salary_grade_ext` (외부 테이블) 의 상세 정보를 각각 조회하고, `Table Type` 필드 값을 기록하여 두 테이블의 차이점을 생각해보세요.

**练习 5 : ** 使用 `DESC FORMATTED` 命令, 分别查看 `dept_info` (内部表) 和 `salary_grade_ext` (外部表) 的详细信息, 记录 `Table Type` 字段的值, 思考两者的区别。

실습 4-3 : HiveQL 파티션 테이블 조작 (45 분)

학습 목표

- Hive 파티션 테이블의 개념과 역할 이해
- 파티션 테이블의 생성, 조회, 추가, 이름 변경, 이동, 삭제 조작 습득
- 정적 파티션 사용 방법 이해

파티션 테이블 개념 복습

> 💡 **왜 파티션 테이블이 필요한가?**

> - Hive 는 기본적으로 전체 테이블 스캔을 수행하여 데이터양이 많으면 효율이 저하됩니다

> - 파티션 테이블은 파티션 필드에 따라 데이터를 여러 하위 디렉터리로 분할합니다

> - 예 : 판매 데이터를 연도/월별로 파티션하면 특정 월 조회 시 해당 파티션 디렉터리만 스캔합니다

실습 4-3-1 : 파티션 테이블 생성

**실습 내용 : ** 연도와 월을 파티션 필드로 하는 판매 데이터 파티션 테이블을 생성합니다.

```sql
-- 2 단계 파티션 테이블 생성 (연도와 월로 파티션)
CREATE TABLE IF NOT EXISTS unit4db.sales_partitioned
(
  sale_id    STRING COMMENT '판매 기록 ID',
  emp_id     INT    COMMENT '직원 ID',
  prod_id    STRING COMMENT '제품 ID',
```

任务 4-3 : HiveQL 操作分区表 (45 分钟)

学习目标

- 理解 Hive 分区表的概念和作用
- 掌握分区表的创建、查看、添加、重命名、移动和删除操作
- 理解静态分区的使用方式

分区表概念回顾

> 💡 **为什么需要分区表？**

> - Hive 默认进行全表扫描，当数据量很大时效率低下

> - 分区表将数据按分区字段分成多个子目录，查询时只扫描指定分区

> - 例如：将销售数据按年份和月份分区，查询特定月份时只扫描对应分区目录

实习 4-3-1 : 创建分区表

任务描述： 创建销售数据分区表，以年份和月份作为分区字段。

```sql
-- 创建二级分区表（按年份和月份分区）
CREATE TABLE IF NOT EXISTS unit4db.sales_partitioned
(
  sale_id    STRING COMMENT '销售记录 ID',
  emp_id     INT    COMMENT '员工 ID',
  prod_id    STRING COMMENT '产品 ID',
```

```
  amount    DOUBLE COMMENT '판매 금액',
  sale_date STRING COMMENT '판매 일자'
)
COMMENT '판매 데이터 파티션 테이블'
PARTITIONED BY (
  sale_year  STRING COMMENT '판매 연도',
  sale_month STRING COMMENT '판매 월'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

> ⚠️ **중요 주의사항 :**

> 1. 파티션 필드명 (`sale_year`, `sale_month`) 은 테이블의 일반 컬럼명과 중복될 수 없습니다

> 2. 파티션 필드는 테이블 생성 후 수정하거나 추가할 수 없습니다

실습 4-3-2 : 파티션 정보 조회

```sql
-- 초기 상태에서 파티션이 없음을 확인
SHOW PARTITIONS unit4db.sales_partitioned;

-- 테이블 상세 정보 조회
DESC FORMATTED unit4db.sales_partitioned;
```

실습 4-3-3 : 파티션 추가

(1) 수동으로 빈 파티션 추가

```sql
-- 2023 년 1 월 파티션 추가
ALTER TABLE unit4db.sales_partitioned
ADD PARTITION (sale_year='2023', sale_month='01')
LOCATION
'/user/hive/warehouse/unit4db/sales_partitioned/year=2023/month=01';
```

```
  amount    DOUBLE COMMENT '销售金额',
  sale_date STRING COMMENT '销售日期'
)
COMMENT '销售数据分区表'
PARTITIONED BY (
  sale_year  STRING COMMENT '销售年份',
  sale_month STRING COMMENT '销售月份'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

> ⚠️ **重要注意事项 :**

> 1. 分区字段名称 (`sale_year`, `sale_month`) 不能与表的普通列名重复

> 2. 分区字段一旦创建，无法修改或添加

实习 4-3-2 : 查看分区信息

```sql
-- 查看 sales_partitioned 表的所有分区 (初始状态为空)
SHOW PARTITIONS unit4db.sales_partitioned;

-- 查看表的详细信息
DESC FORMATTED unit4db.sales_partitioned;
```

实习 4-3-3 : 添加分区

(1) 手动添加空分区

```sql
-- 添加 2023 年 1 月 分区
ALTER TABLE unit4db.sales_partitioned
ADD PARTITION (sale_year='2023', sale_month='01')
LOCATION
'/user/hive/warehouse/unit4db/sales_partitioned/year=2023/month=01';
```

```
-- 2023 년 2 월 파티션 추가
ALTER TABLE unit4db.sales_partitioned
ADD PARTITION (sale_year='2023', sale_month='02')
LOCATION
'/user/hive/warehouse/unit4db/sales_partitioned/year=2023/month=02';

-- 여러 파티션 한번에 추가
ALTER TABLE unit4db.sales_partitioned
ADD IF NOT EXISTS
PARTITION (sale_year='2023', sale_month='03')
PARTITION (sale_year='2023', sale_month='04')
PARTITION (sale_year='2023', sale_month='05')
PARTITION (sale_year='2023', sale_month='06');
```

```sql
-- 파티션 추가 확인
SHOW PARTITIONS unit4db.sales_partitioned;
```

（2）파티션에 데이터 로드

월별 데이터 파일 준비（Linux 터미널）：

```bash
# 월별로 분할된 데이터 파일 생성
grep "2023-01" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_01.txt
grep "2023-02" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_02.txt
grep "2023-03" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_03.txt
grep "2023-04" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_04.txt
grep "2023-05" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_05.txt
grep "2023-06" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_06.txt
```

beeline 에서 데이터 로드：

```sql
-- 지정 파티션에 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/unit4/sales_2023_01.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='01');

LOAD DATA LOCAL INPATH
'/usr/datas/unit4/sales_2023_02.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='02');
```

```
-- 添加 2023 年 2 月 分区
ALTER TABLE unit4db.sales_partitioned
ADD PARTITION (sale_year='2023', sale_month='02')
LOCATION
'/user/hive/warehouse/unit4db/sales_partitioned/year=2023/month=02';

-- 同时添加多个分区
ALTER TABLE unit4db.sales_partitioned
ADD IF NOT EXISTS
PARTITION (sale_year='2023', sale_month='03')
PARTITION (sale_year='2023', sale_month='04')
PARTITION (sale_year='2023', sale_month='05')
PARTITION (sale_year='2023', sale_month='06');
```

```sql
-- 确认分区添加成功
SHOW PARTITIONS unit4db.sales_partitioned;
```

（2）向分区加载数据

首先，准备分月份数据文件：

```bash
# 在 Linux 终端创建按月份分割的数据文件
grep "2023-01" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_01.txt
grep "2023-02" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_02.txt
grep "2023-03" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_03.txt
grep "2023-04" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_04.txt
grep "2023-05" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_05.txt
grep "2023-06" /usr/datas/unit4/sales.txt > /usr/datas/unit4/sales_2023_06.txt
```

在 beeline 中加载数据：

```sql
-- 向指定分区加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/unit4/sales_2023_01.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='01');

LOAD DATA LOCAL INPATH
'/usr/datas/unit4/sales_2023_02.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='02');
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit4/sales_2023_03.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='03');
```

```sql
-- 파티션 데이터 확인
SELECT * FROM unit4db.sales_partitioned WHERE
sale_year='2023' AND sale_month='01';
SELECT COUNT(*) FROM unit4db.sales_partitioned;
```

실습 4-3-4 : 파티션 이름 변경

```sql
-- 2023 년 01 월 파티션을 1 월로 이름 변경 (형식 통일)
ALTER TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='01')
RENAME TO PARTITION (sale_year='2023',
sale_month='1');

-- 결과 확인
SHOW PARTITIONS unit4db.sales_partitioned;
```

실습 4-3-5 : 두 번째 파티션 테이블 생성 및 파티션 이동

（1）동일 구조의 파티션 테이블 `sales_archive` 생성

```sql
CREATE TABLE IF NOT EXISTS unit4db.sales_archive (
  sale_id   STRING COMMENT '판매 기록 ID',
  emp_id    INT    COMMENT '직원 ID',
  prod_id   STRING COMMENT '제품 ID',
  amount    DOUBLE COMMENT '판매 금액',
  sale_date STRING COMMENT '판매 일자'
)
COMMENT '판매 아카이브 파티션 테이블'
PARTITIONED BY (
  sale_year  STRING COMMENT '판매 연도',
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit4/sales_2023_03.txt'
INTO TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='03');
```

```sql
-- 验证分区数据
SELECT * FROM unit4db.sales_partitioned WHERE
sale_year='2023' AND sale_month='01';
SELECT COUNT(*) FROM unit4db.sales_partitioned;
```

实习 4-3-4 : 重命名分区

```sql
-- 将 2023 年 01 月 分区重命名为 2023 年 1 月（格式统一）
ALTER TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='01')
RENAME TO PARTITION (sale_year='2023',
sale_month='1');

-- 验证重命名结果
SHOW PARTITIONS unit4db.sales_partitioned;
```

实习 4-3-5 : 创建第二张分区表并移动分区

（1）创建结构相同的分区表 `sales_archive`

```sql
CREATE TABLE IF NOT EXISTS unit4db.sales_archive (
  sale_id   STRING COMMENT '销售记录 ID',
  emp_id    INT    COMMENT '员工 ID',
  prod_id   STRING COMMENT '产品 ID',
  amount    DOUBLE COMMENT '销售金额',
  sale_date STRING COMMENT '销售日期'
)
COMMENT '销售归档分区表'
PARTITIONED BY (
  sale_year  STRING COMMENT '销售年份',
```

```
  sale_month STRING COMMENT '판매 월'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

（2）파티션 이동（sales_partitioned 의 파티션을 sales_archive 로 이동）

```sql
-- 2023 년 6 월 파티션을 sales_partitioned 에서 sales_archive 로 이동
ALTER TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='06')
RENAME TO PARTITION (sale_year='2023', sale_month='06');

-- 주의 : 실제 이동 작업은 exchange partition 구문을 사용합니다
ALTER TABLE unit4db.sales_archive
EXCHANGE PARTITION (sale_year='2023', sale_month='06')
WITH TABLE unit4db.sales_partitioned;

-- 이동 결과 확인
SHOW PARTITIONS unit4db.sales_partitioned;
SHOW PARTITIONS unit4db.sales_archive;
```

실습 4-3-6 : 파티션 삭제

```sql
-- sales_partitioned 에서 2023 년 04 월 파티션 삭제
ALTER TABLE unit4db.sales_partitioned
DROP IF EXISTS PARTITION (sale_year='2023', sale_month='04');

-- 결과 확인
SHOW PARTITIONS unit4db.sales_partitioned;
```

> 💡 **생각해보기 : ** 파티션을 삭제하면 HDFS 의 해당 디렉터리에 있는 데이터 파일은 여전히 존재할까요 ?

```
  sale_month STRING COMMENT '销售月份'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

（2）移动分区（将 sales_partitioned 的某分区移动到 sales_archive）

```sql
-- 将 2023 年 6 月分区从 sales_partitioned 移动到 sales_archive
ALTER TABLE unit4db.sales_partitioned
PARTITION (sale_year='2023', sale_month='06')
RENAME TO PARTITION (sale_year='2023', sale_month='06');

-- 注意 : 实际移动操作需通过 exchange partition 语法

ALTER TABLE unit4db.sales_archive
EXCHANGE PARTITION (sale_year='2023', sale_month='06')
WITH TABLE unit4db.sales_partitioned;

-- 验证移动结果
SHOW PARTITIONS unit4db.sales_partitioned;
SHOW PARTITIONS unit4db.sales_archive;
```

实习 4-3-6 : 删除分区

```sql
-- 删除 sales_partitioned 中的 2023 年 04 月 分区
ALTER TABLE unit4db.sales_partitioned
DROP IF EXISTS PARTITION (sale_year='2023', sale_month='04');

-- 验证删除结果
SHOW PARTITIONS unit4db.sales_partitioned;
```

> 💡 **思考 : ** 删除分区后，HDFS 上对应目录中的数据文件是否还存在 ?

연습문제 4-3

**연습 1 : ** 직원 출근 기록 파티션 테이블 `attendance`를 생성하세요. 필드 :

- `att_id`（INT）, `emp_id`（INT）, `att_status`（STRING, 출근 상태 : 출근/휴가/출장）
- 파티션 필드 : `att_year`（STRING）, `att_month`（STRING）, `att_day`（STRING）

**연습 2 : ** `attendance` 테이블에 아래 파티션을 추가하고 확인하세요 :

- 파티션 1 : `att_year='2024'`, `att_month='01'`, `att_day='15'`
- 파티션 2 : `att_year='2024'`, `att_month='01'`, `att_day='16'`
- 파티션 3 : `att_year='2024'`, `att_month='02'`, `att_day='01'`

**연습 3 : ** `attendance` 테이블에서 `att_year='2024'`이고 `att_month='01'`인 파티션 정보만 조회하세요.

```sql
SHOW PARTITIONS attendance
PARTITION(att_year='2024', att_month='01');
```

**연습 4 : ** `att_year='2024'`, `att_month='02'`인 파티션을 삭제하고 결과를 확인하세요.

실습 4-4 : HiveQL 버킷 테이블 조작（35 분）

학습 목표

练习题 4-3

**练习 1 : ** 创建员工出勤记录分区表 `attendance`，字段包括 :

- `att_id`（INT）, `emp_id`（INT）, `att_status`（STRING, 出勤状态 : 出勤/请假/出差）
- 分区字段 : `att_year`（STRING）, `att_month`（STRING）, `att_day`（STRING）

**练习 2 : ** 为 `attendance` 表添加以下分区，并验证 :

- 分区 1 : `att_year='2024'`, `att_month='01'`, `att_day='15'`
- 分区 2 : `att_year='2024'`, `att_month='01'`, `att_day='16'`
- 分区 3 : `att_year='2024'`, `att_month='02'`, `att_day='01'`

**练习 3 : ** 查询 `attendance` 表，只查询 `att_year='2024'` 且 `att_month='01'` 的所有分区信息。

```sql
SHOW PARTITIONS attendance
PARTITION(att_year='2024', att_month='01');
```

**练习 4 : ** 删除 `att_year='2024'`, `att_month='02'` 的分区，并验证删除结果。

任务 4-4 : HiveQL 操作分桶表（35 分钟）

学习目标

- 버킷 테이블의 개념과 용도 이해
- 버킷 테이블 생성 및 데이터 로드 방법 습득
- 데이터 쏠림（Skew） 상황에서 버킷의 최적화 효과 이해

버킷 테이블 개념 복습

> 💡 **버킷 vs 파티션：**
> - **파티션**：특정 필드의 값에 따라 데이터를 다른 디렉터리에 저장
> - **버킷**：특정 필드의 해시값에 따라 데이터를 고정된 수의 파일（버킷）에 균등 분배
> - **해결 문제**：버킷은 파티션에서 발생할 수 있는 데이터 쏠림（데이터 불균형） 문제를 해결

실습 4-4-1：버킷 테이블 생성

실습 내용： 성（省份） 필드를 기준으로 데이터를 4 개의 버킷으로 나누는 주문 버킷 테이블을 생성합니다.

1 단계：강제 버킷 설정 활성화

```sql
-- 강제 버킷 활성화（Hive 3.x 는 기본적으로 지원하지만 명시적으로 설정 권장）
SET hive.enforce.bucketing = true;
```

2 단계：버킷 테이블 생성

```sql
CREATE TABLE IF NOT EXISTS unit4db.orders_bucketed (
  order_id  STRING COMMENT '주문 ID',
  cust_name STRING COMMENT '고객명',
```

- 理解分桶表的概念和用途
- 掌握分桶表的创建和数据加载方法
- 了解分桶在数据倾斜场景下的优化作用

分桶表概念回顾

> 💡 **分桶 vs 分区：**
> - **分区**：按某个字段的值将数据存储到不同目录
> - **分桶**：按某个字段的哈希值将数据均匀分配到固定数量的文件（桶）中
> - **解决问题**：分桶解决分区可能带来的数据倾斜问题

实习 4-4-1：创建分桶表

任务描述： 创建订单分桶表，根据省份字段将数据分成 4 个桶。

步骤 1：开启强制分桶配置

```sql
-- 开启强制分桶（Hive 3.x 通常默认支持，但建议明确设置）
SET hive.enforce.bucketing = true;
```

步骤 2：创建分桶表

```sql
CREATE TABLE IF NOT EXISTS unit4db.orders_bucketed (
  order_id  STRING COMMENT '订单 ID',
  cust_name STRING COMMENT '客户名称',
```

```
  province  STRING COMMENT '성 (省份) ',
  category  STRING COMMENT '제품 카테고리',
  amount    DOUBLE COMMENT '주문 금액'
)
COMMENT '주문 버킷 테이블 (성별 4 버킷) '
CLUSTERED BY (province) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

```sql
-- 버킷 내 정렬 기능이 있는 버킷 테이블 생성 (버킷 내 금액 내림차순 정렬)
CREATE TABLE IF NOT EXISTS unit4db.orders_bucketed_sorted (
  order_id  STRING COMMENT '주문 ID',
  cust_name STRING COMMENT '고객명',
  province  STRING COMMENT '성 (省份) ',
  category  STRING COMMENT '제품 카테고리',
  amount    DOUBLE COMMENT '주문 금액'
)
COMMENT '주문 버킷+정렬 테이블'
CLUSTERED BY (province) SORTED BY (amount DESC) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

> 📝 **버킷 규칙 설명 : **

> - 버킷 필드의 해시값을 버킷 수로 나눈 나머지가 같은 데이터가 같은 버킷에 들어갑니다

> - `INT` 타입 : `hash_function(int) = int 값 자체`

> - 기타 타입 (STRING 등) : Java `hashCode()` 메서드 사용

실습 4-4-2 : 버킷 테이블에 데이터 로드

> ⚠️ **중요 : ** 버킷 테이블에는 `LOAD DATA`를 직접 사용할 수 없습니다. `INSERT INTO ... SELECT` 방식이나 임시 테이블을 통해 데이터를 넣어야 합니다.

```
  province  STRING COMMENT '省份',
  category  STRING COMMENT '产品类别',
  amount    DOUBLE COMMENT '订单金额'
)
COMMENT '订单分桶表 (按省份分 4 桶) '
CLUSTERED BY (province) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

```sql
-- 创建带排序的分桶表 (分桶内按金额降序排列)
CREATE TABLE IF NOT EXISTS unit4db.orders_bucketed_sorted (
  order_id  STRING COMMENT '订单 ID',
  cust_name STRING COMMENT '客户名称',
  province  STRING COMMENT '省份',
  category  STRING COMMENT '产品类别',
  amount    DOUBLE COMMENT '订单金额'
)
COMMENT '订单分桶排序表'
CLUSTERED BY (province) SORTED BY (amount DESC) INTO 4 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

> 📝 **分桶规则说明 : **

> - 分桶字段的哈希值对桶数取模，结果相同的数据进入同一个桶

> - `INT` 类型 : `hash_function(int) = int 本身`

> - 其他类型 (STRING 等) : 使用 Java `hashCode()` 方法

实习 4-4-2 : 向分桶表加载数据

> ⚠️ **重要 : ** 分桶表不能直接用 `LOAD DATA` 加载，必须通过 INSERT INTO ... SELECT 方式，或先创建临时表再 INSERT。

1 단계 : 임시 중간 테이블 생성

```sql
CREATE TABLE IF NOT EXISTS unit4db.orders_temp (
  order_id  STRING,
  cust_name STRING,
  province  STRING,
  category  STRING,
  amount    DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

2 단계 : 임시 테이블에 데이터 로드

```sql
LOAD DATA LOCAL INPATH '/usr/datas/unit4/orders.txt'
INTO TABLE unit4db.orders_temp;

-- 데이터 로드 확인
SELECT * FROM unit4db.orders_temp;
```

3 단계 : INSERT INTO ... SELECT 로 버킷 테이블에 데이터 삽입

```sql
-- MapReduce 작업 활성화 (일부 Hive 버전에서 필요)
SET hive.execution.engine = mr;

INSERT INTO TABLE unit4db.orders_bucketed
SELECT order_id, cust_name, province, category,
amount
FROM unit4db.orders_temp;
```

4 단계 : 버킷 효과 확인

```sql
-- 버킷 테이블 데이터 조회
SELECT * FROM unit4db.orders_bucketed;

-- 성별 집계 (버킷 분배 상황 관찰)
SELECT province, COUNT(*) AS cnt FROM
unit4db.orders_bucketed GROUP BY province;
```

```bash
# Linux 터미널에서 HDFS 의 버킷 파일 확인 (4 개 파일이
```

步骤 1：创建临时中间表

```sql
CREATE TABLE IF NOT EXISTS unit4db.orders_temp (
  order_id  STRING,
  cust_name STRING,
  province  STRING,
  category  STRING,
  amount    DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

步骤 2：将数据加载到临时表

```sql
LOAD DATA LOCAL INPATH '/usr/datas/unit4/orders.txt'
INTO TABLE unit4db.orders_temp;

-- 验证数据加载
SELECT * FROM unit4db.orders_temp;
```

步骤 3：通过 INSERT INTO ... SELECT 将数据写入分桶表

```sql
-- 开启 MapReduce 任务（某些 Hive 版本需要此设置）
SET hive.execution.engine = mr;

INSERT INTO TABLE unit4db.orders_bucketed
SELECT order_id, cust_name, province, category,
amount
FROM unit4db.orders_temp;
```

步骤 4：验证分桶效果

```sql
-- 查看分桶表数据
SELECT * FROM unit4db.orders_bucketed;

-- 按省份统计（观察各桶分配情况）
SELECT province, COUNT(*) AS cnt FROM
unit4db.orders_bucketed GROUP BY province;
```

```bash
# 在 Linux 终端查看 HDFS 上的分桶文件（应有 4 个文件）
```

있어야 함)
hdfs dfs -ls /user/hive/warehouse/unit4db/orders_bucketed/
```

---

### 실습 4-4-3 : 버킷 테이블 정보 조회

```sql
-- 버킷 테이블의 상세 메타데이터 정보 조회
DESC FORMATTED unit4db.orders_bucketed;
```

> 💡 **관찰 : ** DESC FORMATTED 출력에서 `Num Buckets`（버킷 수）와 `Bucket Columns`（버킷 필드）를 찾아 테이블 생성 시 설정한 값과 일치하는지 확인하세요.

연습문제 4-4

**연습 1 : ** 직원 급여 버킷 테이블 `emp_salary_bucketed`를 생성하세요. 조건 :

- 필드 : `emp_id`（INT）, `emp_name`（STRING）, `dept`（STRING）, `salary`（DOUBLE）
- `dept`（부서）를 기준으로 **3 개 버킷**으로 분할
- 버킷 내 `salary` 오름차순 정렬

**연습 2 : ** 먼저 임시 테이블 `emp_salary_temp`를 생성하여 `employees.txt` 데이터를 로드한 후, `INSERT INTO ... SELECT`로 `emp_salary_bucketed`에 데이터를 삽입하세요.

**연습 3 : ** `DESC FORMATTED`로 버킷 테이블의 `Num Buckets`가 3 이고 `Bucket Columns`가 `dept`인지 확인하세요.

hdfs dfs -ls /user/hive/warehouse/unit4db/orders_bucketed/
```

实习 4-4-3 : 查看分桶表信息

```sql
-- 查看分桶表的详细元数据信息
DESC FORMATTED unit4db.orders_bucketed;
```

> 💡 **观察 : ** 在 DESC FORMATTED 的输出中，找到 `Num Buckets`（桶数）和 `Bucket Columns`（分桶字段），确认与建表时设置一致。

练习题 4-4

**练习 1 : ** 创建员工薪资分桶表 `emp_salary_bucketed`，要求 :

- 字段 : `emp_id`（INT），`emp_name`（STRING），`dept`（STRING），`salary`（DOUBLE）
- 按 `dept`（部门）分为 **3 个桶**
- 桶内按 `salary` 升序排列

**练习 2 : ** 先创建临时表 `emp_salary_temp` 加载 `employees.txt` 数据，然后通过 INSERT INTO ... SELECT 将数据写入 `emp_salary_bucketed`。

**练习 3 : ** 通过 `DESC FORMATTED` 验证分桶表的 `Num Buckets` 为 3，`Bucket Columns` 为 `dept`。

**연습 4 : ** 쿼리를 실행하여 각 부서의 직원 수를 집계하고 버킷이 균등하게 분배되었는지 관찰하세요.

```sql
SELECT dept, COUNT(*) AS emp_count FROM emp_salary_bucketed GROUP BY dept;
```

실습 4-5 : HiveQL 뷰(View) 조작 (30 분)

학습 목표

- Hive 뷰의 개념과 특징 이해
- 뷰의 생성, 조회, 수정, 삭제 조작 습득

뷰 개념 복습

> 💡 **Hive 뷰 특징 : **
> - 뷰는 가상 테이블로 쿼리 정의만 저장하고 데이터는 저장하지 않습니다
> - 뷰는 복잡한 쿼리 구문을 단순화할 수 있습니다
> - 뷰는 쿼리 성능을 향상시키지 않으며 데이터를 캐시하지 않습니다
> - 기반 테이블이 삭제/수정되면 뷰가 무효화될 수 있습니다

실습 4-5-1 : 뷰 생성

**실습 내용 : ** `employees` 테이블을 기반으로 기술부 직원 뷰를 생성합니다.

**练习 4 : ** 执行查询，统计每个部门的员工数量，观察分桶是否均匀。

```sql
SELECT dept, COUNT(*) AS emp_count FROM emp_salary_bucketed GROUP BY dept;
```

任务 4-5 : HiveQL 操作视图（30 分钟）

学习目标

- 理解 Hive 视图的概念和特点
- 掌握视图的创建、查看、修改和删除操作

视图概念回顾

> 💡 **Hive 视图特点 : **
> - 视图是虚拟表，只保存查询定义，不存储数据
> - 视图可以简化复杂查询语句
> - 视图不提高查询性能，也不缓存数据
> - 基础表结构变更（删除/修改）可能导致视图失效

实习 4-5-1 : 创建视图

**任务描述 : ** 基于 `employees` 表创建技术部员工视图。

```sql
-- employees 테이블에 데이터가 있는지 확인
USE unit4db;
SELECT COUNT(*) FROM employees;
```

(1) 단순 뷰 생성

```sql
-- 기술부 직원 뷰 생성
CREATE VIEW IF NOT EXISTS unit4db.v_tech_employees
AS
SELECT emp_id, emp_name, salary, hire_date
FROM unit4db.employees
WHERE dept = '技术部';
```

(2) 컬럼 주석이 있는 뷰 생성

```sql
-- 고급 직원 뷰 생성 (급여 > 15000), 컬럼에 주석 추가
CREATE VIEW IF NOT EXISTS unit4db.v_high_salary_emp
(
  employee_id    COMMENT '직원 번호',
  employee_name  COMMENT '직원 이름',
  department     COMMENT '소속 부서',
  monthly_salary COMMENT '월급'
)
COMMENT '월급 15000 초과 직원 뷰'
AS
SELECT emp_id, emp_name, dept, salary
FROM unit4db.employees
WHERE salary > 15000;
```

(3) 집계 통계 뷰 생성

```sql
-- 부서별 급여 통계 뷰 생성
CREATE VIEW IF NOT EXISTS
unit4db.v_dept_salary_stats
COMMENT '부서 급여 통계 뷰'
AS
SELECT
  dept          AS department,
  COUNT(emp_id) AS emp_count,
  AVG(salary)   AS avg_salary,
  MAX(salary)   AS max_salary,
  MIN(salary)   AS min_salary
FROM unit4db.employees
```

```sql
-- 确保 employees 表有数据
USE unit4db;
SELECT COUNT(*) FROM employees;
```

(1) 创建简单视图

```sql
-- 创建技术部员工视图
CREATE VIEW IF NOT EXISTS unit4db.v_tech_employees
AS
SELECT emp_id, emp_name, salary, hire_date
FROM unit4db.employees
WHERE dept = '技术部';
```

(2) 创建包含列注释的视图

```sql
-- 创建高薪员工视图（薪资>15000），并为列添加注释
CREATE VIEW IF NOT EXISTS unit4db.v_high_salary_emp
(
  employee_id    COMMENT '员工编号',
  employee_name COMMENT '员工姓名',
  department     COMMENT '所在部门',
  monthly_salary COMMENT '月薪'
)
COMMENT '月薪超过 15000 元的员工视图'
AS
SELECT emp_id, emp_name, dept, salary
FROM unit4db.employees
WHERE salary > 15000;
```

(3) 创建聚合统计视图

```sql
-- 创建部门薪资统计视图
CREATE VIEW IF NOT EXISTS
unit4db.v_dept_salary_stats
COMMENT '部门薪资统计视图'
AS
SELECT
  dept          AS department,
  COUNT(emp_id) AS emp_count,
  AVG(salary)   AS avg_salary,
  MAX(salary)   AS max_salary,
  MIN(salary)   AS min_salary
FROM unit4db.employees
```

```
GROUP BY dept;
```

실습 4-5-2 : 뷰 조회

```sql
-- 1. 현재 데이터베이스의 모든 뷰 조회
SHOW VIEWS IN unit4db;
```

```sql
-- 2. 와일드카드 조회 : "v_"로 시작하는 뷰 조회
SHOW VIEWS IN unit4db LIKE 'v_*';
```

```sql
-- 3. 뷰 기본 구조 정보 조회
DESC unit4db.v_tech_employees;
```

```sql
-- 4. 뷰 상세 정보 조회 (뷰 정의 포함)
DESC FORMATTED unit4db.v_high_salary_emp;
```

실습 4-5-3 : 뷰 데이터 조회

```sql
-- 기술부 직원 뷰 조회
SELECT * FROM unit4db.v_tech_employees;
```

```sql
-- 고급 직원 뷰 조회, 급여 내림차순 정렬
SELECT * FROM unit4db.v_high_salary_emp
ORDER BY monthly_salary DESC;
```

```sql
-- 부서 통계 뷰 조회
SELECT * FROM unit4db.v_dept_salary_stats;
```

```
GROUP BY dept;
```

实习 4-5-2 : 查看视图

```sql
-- 1. 查看当前数据库中所有视图
SHOW VIEWS IN unit4db;
```

```sql
-- 2. 模糊查询 : 查看以 "v_" 开头的视图
SHOW VIEWS IN unit4db LIKE 'v_*';
```

```sql
-- 3. 查看视图结构基本信息
DESC unit4db.v_tech_employees;
```

```sql
-- 4. 查看视图详细信息 (含视图定义语句)
DESC FORMATTED unit4db.v_high_salary_emp;
```

实习 4-5-3 : 查询视图数据

```sql
-- 查询技术部员工视图
SELECT * FROM unit4db.v_tech_employees;
```

```sql
-- 查询高薪员工视图, 并按薪资降序排列
SELECT * FROM unit4db.v_high_salary_emp
ORDER BY monthly_salary DESC;
```

```sql
-- 查询部门统计视图
SELECT * FROM unit4db.v_dept_salary_stats;
```

실습 4-5-4 : 뷰 수정

(1) 뷰 속성 수정

```sql
ALTER VIEW unit4db.v_tech_employees
SET TBLPROPERTIES ('last_modified' = '2024-06-01',
'status' = 'active');
```

(2) 뷰 쿼리 정의 수정

```sql
-- v_tech_employees 뷰 수정 : 기술부 한정 조건 제거, 전체 부서 중 급여 12000 이상 직원으로 변경
ALTER VIEW unit4db.v_tech_employees
AS
SELECT emp_id, emp_name, dept, salary, hire_date
FROM unit4db.employees
WHERE salary >= 12000;

-- 수정 결과 확인
SELECT * FROM unit4db.v_tech_employees;
DESC FORMATTED unit4db.v_tech_employees;
```

실습 4-5-5 : 뷰 삭제

```sql
-- 뷰 삭제
DROP VIEW IF EXISTS unit4db.v_tech_employees;
DROP VIEW IF EXISTS unit4db.v_high_salary_emp;

-- 삭제 확인
SHOW VIEWS IN unit4db;
```

연습문제 4-5

**연습 1 : ** `employees` 테이블을 기반으로 뷰

实习 4-5-4 : 修改视图

(1) 修改视图属性

```sql
ALTER VIEW unit4db.v_tech_employees
SET TBLPROPERTIES ('last_modified' = '2024-06-01',
'status' = 'active');
```

(2) 修改视图的查询定义

```sql
-- 修改 v_tech_employees 视图 : 包含所有部门员工, 不再限制技术部
ALTER VIEW unit4db.v_tech_employees
AS
SELECT emp_id, emp_name, dept, salary, hire_date
FROM unit4db.employees
WHERE salary >= 12000;

-- 验证修改结果
SELECT * FROM unit4db.v_tech_employees;
DESC FORMATTED unit4db.v_tech_employees;
```

实习 4-5-5 : 删除视图

```sql
-- 删除视图
DROP VIEW IF EXISTS unit4db.v_tech_employees;
DROP VIEW IF EXISTS unit4db.v_high_salary_emp;

-- 验证删除结果
SHOW VIEWS IN unit4db;
```

练习题 4-5

**练习 1 : ** 基于 `employees` 表, 创建视图

`v_dept_emp_list`를 생성하세요. 각 부서 직원 목록을 표시하며, 필드는 부서명, 직원 ID, 직원이름, 급여를 포함합니다. 부서명 오름차순, 급여 내림차순으로 정렬하세요.

**연습 2 : ** `orders_bucketed` 테이블을 기반으로 뷰 `v_province_orders`를 생성하세요. 각 성（省份）의 주문 건수와 총 금액을 통계하세요.

**연습 3 : ** 뷰 `v_dept_emp_list`를 수정하여 부서 평균 급여보다 높은 직원만 표시하는 필터 조건을 추가하세요.

**연습 4 : ** `DESC FORMATTED`로 `v_dept_emp_list`의 뷰 정의를 확인하여 쿼리문이 올바른지 검증하세요.

실습 4-6 : HiveQL 종합 DDL 조작（30 분）

학습 목표

- Hive DDL 조작 복습 및 심화（데이터베이스, 테이블, 파티션, 뷰）
- Hive 인덱스의 기본 개념과 조작 이해
- SHOW 시리즈 구문 종합 활용 습득

실습 4-6-1 : SHOW 구문 종합 활용

```sql
-- 1. 모든 데이터베이스 표시
SHOW DATABASES;
```

`v_dept_emp_list`，显示各部门员工列表，字段包括：部门名称、员工 ID、员工姓名、薪资，并按部门名称升序、薪资降序排列。

**练习 2 : ** 基于 `orders_bucketed` 表，创建视图 `v_province_orders`，统计各省份的订单数量和总金额。

**练习 3 : ** 修改视图 `v_dept_emp_list`，增加过滤条件：只显示薪资高于部门平均薪资的员工。

**练习 4 : ** 使用 `DESC FORMATTED` 查看 `v_dept_emp_list` 的视图定义，确认查询语句是否正确。

任务 4-6 : HiveQL 综合 DDL 操作（30 分钟）

学习目标

- 复习和巩固 Hive DDL 操作（数据库、表、分区、视图）
- 了解 Hive 索引的基本概念和操作
- 掌握 SHOW 系列语法的综合使用

实习 4-6-1 : SHOW 语法综合应用

```sql
-- 1. 显示所有数据库
SHOW DATABASES;
```

```
-- 2. unit4db 의 모든 테이블 표시
USE unit4db;
SHOW TABLES;

-- 3. unit4db 의 모든 뷰 표시
SHOW VIEWS;

-- 4. sales_partitioned 테이블의 파티션 정보 표시
SHOW PARTITIONS sales_partitioned;

-- 5. 테이블 생성 구문 표시
SHOW CREATE TABLE employees;

-- 6. 테이블 컬럼 정보 표시
SHOW COLUMNS IN employees;

-- 7. 함수 목록 표시 (참고용)
SHOW FUNCTIONS;
```

실습 4-6-2 : Hive 인덱스 조작 (참고 사항)

> ⚠️ **주의 : ** Hive 3.x 에서는 인덱스 기능이 폐기되었습니다. 아래 내용은 참고용이며 최신 버전에서는 사용을 권장하지 않습니다. 실제 운영 환경에서는 파티션, ORC 파일 형식의 내장 인덱스 또는 Bloom Filter 를 활용하여 쿼리 성능을 최적화합니다.

```sql
-- 인덱스 구문 참고 (Hive 2.x 이하 버전에서 유효)
/*
CREATE INDEX idx_emp_name
ON TABLE employees (emp_name)
AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
WITH DEFERRED REBUILD
TBLPROPERTIES ('creator' = 'student');
*/

-- Hive 3.x 에서 쿼리 성능 최적화 권장 방법 :
-- 1. 파티션 테이블 사용
```

```
-- 2. 显示 unit4db 中所有表
USE unit4db;
SHOW TABLES;

-- 3. 显示 unit4db 中所有视图
SHOW VIEWS;

-- 4. 显示 sales_partitioned 表的分区信息
SHOW PARTITIONS sales_partitioned;

-- 5. 显示表的创建语句
SHOW CREATE TABLE employees;

-- 6. 显示表列信息
SHOW COLUMNS IN employees;

-- 7. 显示函数列表（了解用）
SHOW FUNCTIONS;
```

实习 4-6-2 : Hive 索引操作（了解）

> ⚠️ **注意 : ** Hive 3.x 中已废弃索引功能，以下内容仅供了解，不建议在新版本中使用。在实际生产中，通常使用分区、ORC 文件格式的内置索引或 Bloom Filter 来优化查询性能。

```sql
-- 了解索引语法（Hive 2.x 及以下版本有效）
/*
CREATE INDEX idx_emp_name
ON TABLE employees (emp_name)
AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
WITH DEFERRED REBUILD
TBLPROPERTIES ('creator' = 'student');
*/

-- 在现代 Hive 3.x 中，优化查询性能的推荐方式 :
-- 1. 使用分区表
```

```
-- 2. ORC 파일 형식 사용 (내장 경량 인덱스 포함)
-- 3. 적절한 버킷 설정

-- 실습 : employees 테이블을 ORC 형식으로 변환 (더 나은 쿼리 성능)
CREATE TABLE IF NOT EXISTS unit4db.employees_orc (
  emp_id     INT    COMMENT '직원 ID',
  emp_name   STRING COMMENT '직원 이름',
  dept       STRING COMMENT '소속 부서',
  salary     DOUBLE COMMENT '급여',
  hire_date STRING COMMENT '입사일'
)
STORED AS ORC
TBLPROPERTIES ('orc.compress'='SNAPPY');

-- employees 테이블에서 ORC 테이블로 데이터 삽입
INSERT INTO unit4db.employees_orc
SELECT * FROM unit4db.employees;

-- 확인
SELECT * FROM unit4db.employees_orc;
```

실습 4-6-3 : 종합 DDL 조작 복습

```sql
-- 1. 이번 실습에서 생성한 모든 객체 조회
USE unit4db;
SHOW TABLES;
SHOW VIEWS;

-- 2. 각 테이블의 행 수 확인
SELECT 'employees' AS table_name, COUNT(*) AS row_count FROM employees
UNION ALL
SELECT 'products_ext', COUNT(*) FROM products_ext
UNION ALL
SELECT 'orders_bucketed', COUNT(*) FROM orders_bucketed
UNION ALL
SELECT 'sales_partitioned', COUNT(*) FROM sales_partitioned;
```

종합 연습문제

```
-- 2. 使用 ORC 文件格式（内置轻量索引）
-- 3. 合理使用分桶

-- 演示：将 employees 表转换为 ORC 格式（更好的查询性能）
CREATE TABLE IF NOT EXISTS unit4db.employees_orc (
  emp_id     INT    COMMENT '员工 ID',
  emp_name   STRING COMMENT '员工姓名',
  dept       STRING COMMENT '所属部门',
  salary     DOUBLE COMMENT '薪资',
  hire_date STRING COMMENT '入职日期'
)
STORED AS ORC
TBLPROPERTIES ('orc.compress'='SNAPPY');

-- 从 employees 表插入数据到 ORC 表
INSERT INTO unit4db.employees_orc
SELECT * FROM unit4db.employees;

-- 验证
SELECT * FROM unit4db.employees_orc;
```

实习 4-6-3：综合 DDL 操作回顾

```sql
-- 1. 查看本次实习创建的所有对象
USE unit4db;
SHOW TABLES;
SHOW VIEWS;

-- 2. 查看各表的行数
SELECT 'employees' AS table_name, COUNT(*) AS row_count FROM employees
UNION ALL
SELECT 'products_ext', COUNT(*) FROM products_ext
UNION ALL
SELECT 'orders_bucketed', COUNT(*) FROM orders_bucketed
UNION ALL
SELECT 'sales_partitioned', COUNT(*) FROM sales_partitioned;
```

综合练习题

종합 연습 (30 분)

다음 요구사항에 따라 직원 관리 데이터베이스의 완전한 DDL 작업을 수행하세요:

요구 1: 데이터베이스 작업

- `hr_db` 데이터베이스 생성, 주석: "인사 데이터베이스", HDFS 경로: `/user/hive/warehouse/hr_db`
- 속성 추가: `dept`=`HR`, `version`=`2.0`
- 속성 수정: `last_update`=`2024-06-01` 추가

요구 2: 테이블 작업

- `hr_db`에 내부 테이블 `hr_employees` 생성. 필드:
  - `emp_id` INT, `emp_name` STRING, `age` INT, `dept_id` INT, `salary` DOUBLE, `city` STRING
  - 필드 구분자: 쉼표
- `hr_db`에 외부 테이블 `hr_departments` 생성. 필드: `dept_id` INT, `dept_name` STRING, `dept_head` STRING
  - LOCATION: `/user/hive/warehouse/hr_db/departments`

요구 3: 파티션 테이블

- `hr_db`에 `hr_salary_history` 파티션 테이블 생성. `year` (STRING) 와 `quarter` (STRING) 로 파티션
- 필드: `emp_id` INT, `base_salary` DOUBLE, `bonus` DOUBLE
- 파티션 추가: `year='2024',quarter='Q1'` 및 `year='2024',quarter='Q2'`

요구 4: 버킷 테이블

- `hr_db`에 버킷 테이블 `hr_employees_bucketed` 생성
- `hr_employees`와 동일한 필드, `dept_id`로 4 버킷 분할, 버킷 내 `salary` 내림차순 정렬

综合练习（30 分钟）

按照以下要求，完成一个完整的员工管理数据库的 DDL 操作：

要求 1：数据库操作

- 创建数据库 `hr_db`，注释为"人力资源数据库"，HDFS 路径为 `/user/hive/warehouse/hr_db`
- 添加属性：`dept`=`HR`, `version`=`2.0`
- 修改属性：添加 `last_update`=`2024-06-01`

要求 2：表操作

- 在 `hr_db` 中创建内部表 `hr_employees`，字段如下：
  - `emp_id` INT, `emp_name` STRING, `age` INT, `dept_id` INT, `salary` DOUBLE, `city` STRING
  - 字段分隔符为逗号
- 在 `hr_db` 中创建外部表 `hr_departments`，字段：`dept_id` INT, `dept_name` STRING, `dept_head` STRING
  - LOCATION: `/user/hive/warehouse/hr_db/departments`

要求 3：分区表

- 在 `hr_db` 中创建分区表 `hr_salary_history`，按 `year`（STRING）和 `quarter`（STRING）分区
- 字段：`emp_id` INT, `base_salary` DOUBLE, `bonus` DOUBLE
- 添加以下分区：year='2024',quarter='Q1' 和 year='2024',quarter='Q2'

要求 4：分桶表

- 在 `hr_db` 中创建分桶表 `hr_employees_bucketed`
- 与 `hr_employees` 字段相同，按 `dept_id` 分为 4 个桶，桶内按 `salary` 降序排列

요구 5 : 뷰

- 뷰 `v_senior_employees` 생성 : 급여 12000 초과 직원 정보 표시 (전체 필드)

- 뷰 `v_city_stats` 생성 : 도시별 직원 수와 평균 급여 통계

要求 5：视图

- 创建视图 `v_senior_employees`：显示工资高于 12000 的员工信息（全字段）

- 创建视图 `v_city_stats`：统计各城市员工数量和平均薪资

요구 6 : 조회 및 검증

- SHOW 명령으로 생성된 모든 객체 확인

- DESC FORMATTED 으로 `hr_employees` (내부 테이블) 와 `hr_departments` (외부 테이블) 의 타입 차이 확인

要求 6：查询验证

- 使用 SHOW 命令验证所有创建的对象

- 使用 DESC FORMATTED 查看 `hr_employees`（内部表）和 `hr_departments`（外部表）的类型区别

📌 본 장 실습 요약

이번 실습을 통해 다음 Hive DDL 핵심 조작을 습득했습니다 :

조작 대상	주요 명령어
데이터베이스	CREATE/SHOW/DESC/USE/ALTER/DROP DATABASE
데이터 테이블	CREATE TABLE (내부/외부) /SHOW TABLES/DESC/ALTER TABLE/DROP TABLE
파티션 테이블	PARTITIONED BY/ADD PARTITION/RENAME/EXCHANGE/DROP PARTITION
버킷 테이블	CLUSTERED BY/SORTED BY/INTO N BUCKETS
뷰	CREATE/SHOW/DESC/ALTER/DROP VIEW
종합	SHOW CREATE TABLE/SHOW COLUMNS/STORED AS ORC

**데이터 로드 방법 요약 : **

```sql
-- 로컬 파일 시스템에서 로드
```

📌 本章实习总结

通过本次实习，我们掌握了以下 Hive DDL 核心操作：

操作对象	关键命令
数据库	CREATE/SHOW/DESC/USE/ALTER/DROP DATABASE
数据表	CREATE TABLE（内部/外部）/SHOW TABLES/DESC/ALTER TABLE/DROP TABLE
分区表	PARTITIONED BY/ADD PARTITION/RENAME/EXCHANGE/DROP PARTITION
分桶表	CLUSTERED BY/SORTED BY/INTO N BUCKETS
视图	CREATE/SHOW/DESC/ALTER/DROP VIEW
综合	SHOW CREATE TABLE/SHOW COLUMNS/STORED AS ORC

数据加载方式：

```sql
-- 从本地文件系统加载
```

```
LOAD DATA LOCAL INPATH '/usr/datas/unit4/파일명.txt'
INTO TABLE 테이블명;

-- 지정 파티션에 로드
LOAD DATA LOCAL INPATH '/usr/datas/unit4/파일명.txt'
INTO TABLE 테이블명 PARTITION (파티션필드='값');

-- 버킷 테이블은 INSERT INTO ... SELECT 로 로드
INSERT INTO TABLE 버킷테이블 SELECT * FROM
임시테이블;
```

---

*본 워크북은 《Hive 데이터웨어하우스 응용 교재》
제 4 장 내용을 기반으로 작성되었으며, 약 4 시간의 수업
실습에 적합합니다.*

*사용 환경：Hadoop 3.3.5 + Apache Hive 3.1.3 + VMware
Ubuntu*

```
LOAD DATA LOCAL INPATH '/usr/datas/unit4/文件名.txt'
INTO TABLE 表名;

-- 加载到指定分区
LOAD DATA LOCAL INPATH '/usr/datas/unit4/文件名.txt'
INTO TABLE 表名 PARTITION (分区字段='值');

-- 分桶表通过 INSERT INTO ... SELECT 加载
INSERT INTO TABLE 分桶表 SELECT * FROM 临时表;
```

---

*本工作手册基于《Hive 数据仓库应用教程》第 4 章内
容编写，适用于约 4 小时的课堂实习。*

*使用环境：Hadoop 3.3.5 + Apache Hive 3.1.3 + VMware
Ubuntu*