데이터베이스와 SQL

3장

쿼리 입문

빅데이터 분석가 과정

목차

3.2 쿼리(query) 절

- select 문
  - select문은 여러 개의 구성 요소 및 절(clause)로 구성

clause(절) 이름	사용 목적
select	쿼리 결과에 포함할 열을 결정
from	데이터를 검색할 테이블과 테이블을 조인하는 방법을 식별
where	불필요한 데이터를 걸러냄 (조건식)
group by	공통 열 값을 기준으로 행을 그룹화함
having	불필요한 그룹을 걸러냄 (필터링)
order by	하나 이상의 열을 기준으로 최종 결과의 행을 정렬

3.3 Select 절

- sakila 데이터베이스 선택
  - language 테이블 구성 (3개의 컬럼)

컬럼명	#	Data Type	Not Null	Comment	Auto Increment	Key
language_id	1	tinyint unsigned	[v]		[v]	PRI
name	2	char(20)	[v]		[ ]	
last_update	3	timestamp	[v]		[ ]	

- select 절
  - 최종 결과셋에 포함할 항목(열)을 결정

```
use sakila;
SELECT * FROM language;
```

  - language 테이블의 전체 목록 보기
  - * (asterisk): 모든 열을 지정

```
language_id¦name    ¦last_update        ¦
-----------+--------+-------------------+
          1¦English ¦2006-02-15 05:02:19¦
          2¦Italian ¦2006-02-15 05:02:19¦
          3¦Japanese¦2006-02-15 05:02:19¦
          4¦Mandarin¦2006-02-15 05:02:19¦
          5¦French  ¦2006-02-15 05:02:19¦
          6¦German  ¦2006-02-15 05:02:19¦
```

3.3 Select 절

- 일부 열(column)만 검색

```
SELECT language_id, name, last_update FROM language;
```

```
language_id¦name    ¦last_update        ¦
-----------+--------+-------------------+
          1¦English ¦2006-02-15 05:02:19¦
          2¦Italian ¦2006-02-15 05:02:19¦
          3¦Japanese¦2006-02-15 05:02:19¦
          4¦Mandarin¦2006-02-15 05:02:19¦
          5¦French  ¦2006-02-15 05:02:19¦
          6¦German  ¦2006-02-15 05:02:19¦
```

```
SELECT name FROM language;
```

```
name    ¦
--------+
English ¦
Italian ¦
Japanese¦
Mandarin¦
French  ¦
German  ¦
```

3.3 Select 절

- Select절에 추가할 수 있는 항목
  - 숫자 또는 문자열
  - 표현식
  - 내장 함수 호출 및 사용자 정의 함수 호출

```
select language_id,
    'COMMON' language_usage,
    language_id * 3.14 lang_pi_value,
    upper(name) language_name
from language;
```

원래 language 테이블에는 없는 컬럼
(가상 컬럼)

language_id	language_usage	lang_pi_value	language_name
1	COMMON	3.14	ENGLISH
2	COMMON	6.28	ITALIAN
3	COMMON	9.42	JAPANESE
4	COMMON	12.56	MANDARIN
5	COMMON	15.70	FRENCH
6	COMMON	18.84	GERMAN

가상 컬럼

3.3 Select 절

- 열의 별칭(column alias)
  - 열의 레이블을 지정할 수 있음
  - 출력을 이해하기 쉽게 함
  - AS(as) 키워드 사용: 가독성 향상

```
select language_id,
    'COMMON' language_usage,
    language_id * 3.14 lang_pi_value,
    upper(name) language_name
from language;
```

열의 별칭

AS 키워드 사용

```
select language_id,
    'COMMON '  AS language_usage,
    language_id * 3.14 AS lang_pi_value,
    upper(name) AS language_name
from language;
```

3.3 Select 절

- 중복 제거

```
select actor_id from film_actor order by actor_id;
```

```
actor_id|
--------+
       1|
       1|
       1|
       1|
       1|
       1|
       1|
       1|
       1|
       1|
 . . .
     200|
```

중복 발생

  - 동일한 배우가 여러 영화에 출연: 중복된 actor_id 발생
    - all 키워드: 기본값, 명시적으로 지정할 필요가 없음
  - distinct 키워드 사용: 중복 제거

```
select distinct actor_id from film_actor order by actor_id;
```

```
actor_id|
--------+
       1|
       2|
       3|
       4|
       5|
       6|
       7|
       8|
       9|
      10|
 . . .
     197|
     198|
     199|
     200|
```

distinct 결과
- 데이터 정렬이 먼저 이루어짐(시간 소요)

3.4 From 절

- From 절 역할
  - 쿼리에 사용되는 테이블을 명시
  - 테이블을 연결하는 수단

- 테이블 유형: from 절에 포함
  - **영구 테이블(permanent table)**
    - create table 문으로 생성된 테이블
    - 실제 데이터베이스에 존재하는 테이블
  - **파생 테이블(derived table)**
    - 하위 쿼리(subquery)에서 반환하고 메모리에 보관된 행
  - **임시 테이블(temporary table)**
    - 메모리에 저장된 휘발성 데이터
  - **가상 테이블(virtual table)**
    - create view문으로 생성

3.4 From 절

- 파생 테이블
  - subquery(서브 쿼리): 9장 참조
    - from 절에 위치한 select문(서브 쿼리)은 실행 결과로 테이블을 생성: 파생 테이블
    - 즉, 다른 테이블과의 상호작용을 할 수 있는 파생 테이블을 생성

```
SELECT ... FROM (subquery) [AS] tbl_name ...
```

```
select concat(cust.last_name, ', ', cust.first_name) full_name
from
    (select first_name, last_name, email
    from customer
    where first_name = 'JESSIE'
    ) as cust;
```

서브 쿼리 (subquery)
- 파생 테이블 cust를 생성

cust: 서브 쿼리의 별칭
- 파생 테이블 이름

```
full_name    ¦
-------------+
BANKS, JESSIE¦
MILAM, JESSIE¦
```

- concat(문자열1, 문자열2, ...): 둘 이상의 문자열을 순서대로 합쳐서 반환

3.4 From 절

- 임시 테이블
  - 휘발성의 테이블: 데이터베이스 세션이 닫힐 때 사라짐

```
create temporary table actors_j
    (actor_id smallint(5),
    first_name varchar(45),
    last_name varchar(45));
desc actors_j;
```

임시테이블 (actors_j) 생성

smallint(5)
- 화면출력시 5자리 맞춤

```
Field     |Type        |Null|Key|Default|Extra|
----------+------------+----+---+-------+-----+
actor_id  |smallint    |YES |   |       |NULL |
first_name|varchar(45)|YES |   |       |NULL |
last_name |varchar(45)|YES |   |       |NULL |
```

```
insert into actors_j
    select actor_id, first_name, last_name
    from actor where last_name like 'J%';
select * from actors_j;
```

actor 테이블에서 last_name이 'J'로 시작하는 데이터를 찾아서 actors_j 임시 테이블에 저장

```
actor_id|first_name|last_name|
--------+----------+---------+
     119|WARREN    |JACKMAN  |
     131|JANE      |JACKMAN  |
       8|MATTHEW   |JOHANSSON|
      64|RAY       |JOHANSSON|
     146|ALBERT    |JOHANSSON|
      82|WOODY     |JOLIE    |
      43|KIRK      |JOVOVICH |
```

3.4 From 절

- **가상 테이블(View)**
  - SQL 쿼리의 결과 셋을 기반으로 만들어진 가상 테이블
  - 실제 데이터가 저장되는 것이 아닌, view를 통해 데이터를 관리
  - 복잡한 쿼리문을 매번 사용하지 않고 가상 테이블로 만들어서 쉽게 접근함

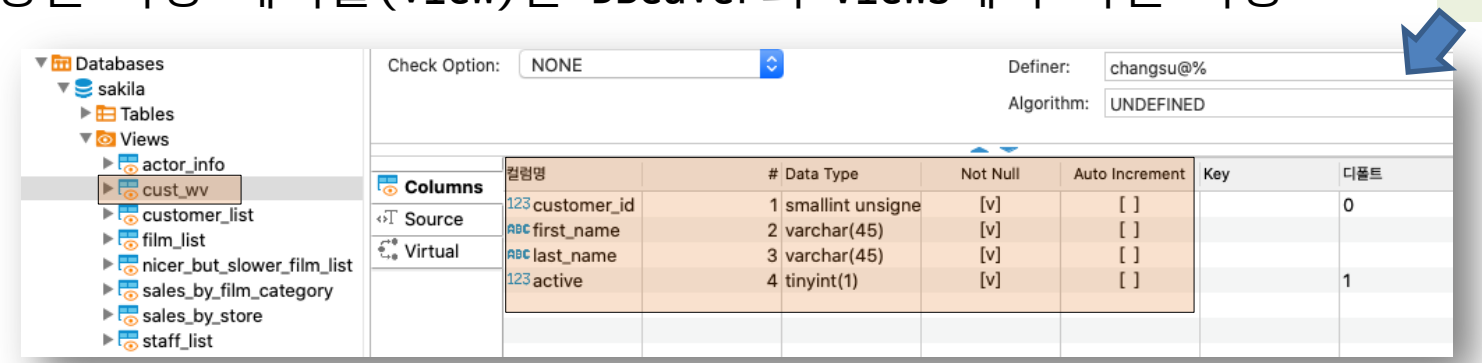
```
CREATE VIEW [뷰이름] AS
SELECT [컬럼명1] ... FROM [테이블명]
```

```
create view cust_vw as
    select customer_id, first_name, last_name, active
    from customer;

select * from cust_vw;
```

```
customer_id|first_name|last_name |active|
-----------+----------+----------+------+
          1|MARY      |SMITH     |     1|
          2|PATRICIA  |JOHNSON   |     1|
          3|LINDA     |WILLIAMS  |     1|
          4|BARBARA   |JONES     |     1|
                        . . .
```

  - 생성된 가상 테이블(view)는 DBeaver의 Views에서 확인 가능

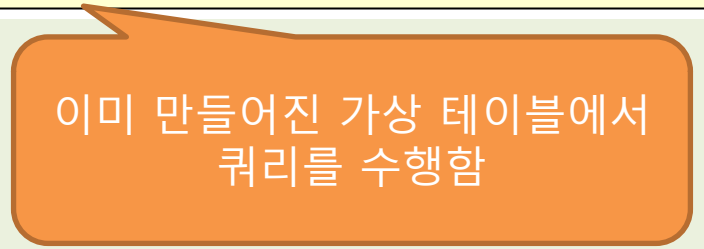


3.4 From 절

- 가상 테이블(view)

```
select first_name, last_name
from cust_vw where active=0;
```

```
first_name¦last_name¦
----------+---------+
SANDRA    ¦MARTIN   ¦
JUDITH    ¦COX      ¦
SHEILA    ¦WELLS    ¦
ERICA     ¦MATTHEWS ¦
HEIDI     ¦LARSON   ¦
PENNY     ¦NEAL     ¦
KENNETH   ¦GOODEN   ¦
HARRY     ¦ARCE     ¦
NATHAN    ¦RUNYON   ¦
THEODORE  ¦CULP     ¦
MAURICE   ¦CRAWLEY  ¦
BEN       ¦EASTER   ¦
CHRISTIAN ¦JUNG     ¦
JIMMIE    ¦EGGLESTON¦
TERRANCE  ¦ROUSH    ¦
```



- 가상 테이블 삭제

```
drop view 뷰이름;
```

3.4 Where 절

- 테이블 연결
  - JOIN(INNER JOIN)
    - 두 개 이상의 테이블을 묶어서 하나의 결과 집합을 만들어 내는 것

```
SELECT <열 목록>
FROM <기준 테이블> [INNER] JOIN <참조할 테이블>
    ON <조인 조건>
[WHERE 검색 조건]
```

```
select customer.first_name, customer.last_name,
    time(rental.rental_date) as rental_time
from customer inner join rental
    on customer.customer_id = rental.customer_id
where date(rental.rental_date) = '2005-06-14';
```

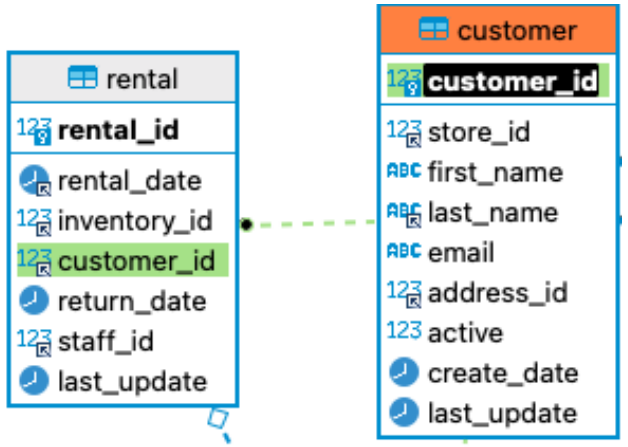
연결할 테이블
(customer, rental 테이블 연결)

연결(Join) 조건

    - 연결(결합) 조건: on
      - customer 테이블의 customer_id와 rental 테이블의 customer_id의 값이 일치하는 경우에만 데이터를 가져옴

3.4 Where 절

- 테이블 연결
  - customer 테이블과 rental 테이블 확인
  - 두 테이블은 customer_id로 연결되어 있음



- DATETIME 데이터
  - date() 함수
    - datetime 데이터에서 date 정보(YYYY-MM-DD)만 추출

```
select date('2021-07-29 09:02:03');
```

```
date('2021-07-29 09:02:03')¦
----------------------------+
                 2021-07-29¦
```

  - time() 함수
    - time 정보(HH:MI:SS) 정보만 추출

```
select time('2021-07-29 09:02:03');
```

```
time('2021-07-29 09:02:03')¦
----------------------------+
                    09:02:03¦
```

3.4 Where 절

- 테이블 연결 (inner join) 결과

```
select customer.first_name, customer.last_name,
    time(rental.rental_date) rental_date
from customer inner join rental
    on customer.customer_id = rental.customer_id
where date(rental.rental_date) = '2005-06-14';
```

```
first_name¦last_name¦rental_date¦
----------+---------+-----------+
JEFFERY   ¦PINSON   ¦   22:53:33¦
ELMER     ¦NOE      ¦   22:55:13¦
MINNIE    ¦ROMERO   ¦   23:00:34¦
MIRIAM    ¦MCKINNEY ¦   23:07:08¦
DANIEL    ¦CABRAL   ¦   23:09:38¦
TERRANCE  ¦ROUSH    ¦   23:12:46¦
JOYCE     ¦EDWARDS  ¦   23:16:26¦
GWENDOLYN ¦MAY      ¦   23:16:27¦
CATHERINE ¦CAMPBELL ¦   23:17:03¦
MATTHEW   ¦MAHAN    ¦   23:25:58¦
HERMAN    ¦DEVORE   ¦   23:35:09¦
AMBER     ¦DIXON    ¦   23:42:56¦
TERRENCE  ¦GUNDERSON¦   23:47:35¦
SONIA     ¦GREGORY  ¦   23:50:11¦
CHARLES   ¦KOWALSKI ¦   23:54:34¦
JEANETTE  ¦GREENE   ¦   23:54:46¦
```

3.4 Where 절

- 테이블 별칭 정의
  - 여러 테이블을 join할 경우, 테이블 및 열 참조 방법
    - 테이블 이름 및 열 이름 사용
    - 각 테이블의 별칭을 할당하고 쿼리 전체에서 해당 별칭을 사용
      - AS 키워드 사용

```
select c.first_name, c.last_name,
    time(r.rental_date) rental_date
from customer c inner join rental r
    on c.customer_id = r.customer_id
where date(r.rental_date) = '2005-06-14';
```

as 키워드 추가

```
select c.first_name, c.last_name,
    time(r.rental_date) as rental_time
from customer as c inner join rental as r
    on c.customer_id = r.customer_id
where date(r.rental_date) = '2005-06-14';
```

```
first_name¦last_name¦rental_time¦
----------+---------+-----------+
JEFFERY   ¦PINSON   ¦   22:53:33¦
ELMER     ¦NOE      ¦   22:55:13¦
MINNIE    ¦ROMERO   ¦   23:00:34¦
MIRIAM    ¦MCKINNEY ¦   23:07:08¦
DANIEL    ¦CABRAL   ¦   23:09:38¦
TERRANCE  ¦ROUSH    ¦   23:12:46¦
JOYCE     ¦EDWARDS  ¦   23:16:26¦
GWENDOLYN ¦MAY      ¦   23:16:27¦
CATHERINE ¦CAMPBELL ¦   23:17:03¦
MATTHEW   ¦MAHAN    ¦   23:25:58¦
HERMAN    ¦DEVORE   ¦   23:35:09¦
AMBER     ¦DIXON    ¦   23:42:56¦
TERRENCE  ¦GUNDERSON¦   23:47:35¦
SONIA     ¦GREGORY  ¦   23:50:11¦
CHARLES   ¦KOWALSKI ¦   23:54:34¦
JEANETTE  ¦GREENE   ¦   23:54:46¦
```

3.5 Where 절

- where 절
  - 필터 조건: 조건에 맞는 행의 데이터만 가져옴
    - AND, OR, NOT 연산자 사용

```
select title
from film
where rating='G' and rental_duration >= 7;
```

    - 필터 조건: film 테이블에서 rating이 G등급이고, rental_duration이 7이상

```
title                 ¦
----------------------+
BLANKET BEVERLY       ¦
BORROWERS BEDAZZLED   ¦
BRIDE INTRIGUE        ¦
CATCH AMISTAD         ¦
CITIZEN SHREK         ¦
COLDBLOODED DARLING   ¦
CONTROL ANTHEM        ¦
CRUELTY UNFORGIVEN    ¦
. . .
WAKE JAWS             ¦
WAR NOTTING           ¦
```

3.5 Where 절

- Where절 검색 조건
  - and, or 사용
    - (G등급이면서 7일 이상 대여할 수 있거나), (PG-13 등급이면서 3일 이내로 대여)할 수 있는 영화 목록

```
select title, rating, rental_duration
from film
where (rating='G' and rental_duration >= 7) or
      (rating='PG-13' and rental_duration < 4);
```

```
title                 |rating|rental_duration|
----------------------+------+---------------+
ALABAMA DEVIL         |PG-13 |              3|
BACKLASH UNDEFEATED   |PG-13 |              3|
BILKO ANONYMOUS       |PG-13 |              3|
BLANKET BEVERLY       |G     |              7|
BORROWERS BEDAZZLED   |G     |              7|
BRIDE INTRIGUE        |G     |              7|
CASPER DRAGONFLY      |PG-13 |              3|
CATCH AMISTAD         |G     |              7|
CITIZEN SHREK         |G     |              7|
COLDBLOODED DARLING   |G     |              7|
CONFUSED CANDLES      |PG-13 |              3|
     .  .  .
WAR NOTTING           |G     |              7|
WORLD LEATHERNECKS    |PG-13 |              3|
```

3.6 Group by절과 having절

- GROUP BY
  - 열(column)의 데이터를 그룹화

```
SELECT 컬럼 FROM 테이블 GROUP BY 그룹화할 컬럼;
```

- HAVING (필터링 조건)
  - 특정 열을 그룹화한 결과에 필터링 조건을 설정: group by 이후에 위치
    - WHERE: 모든 필드에 대한 필터링 수행, from 다음에 위치

```
select c.first_name, c.last_name, count(*) as rental_count
from customer as c
    inner join rental as r
    on c.customer_id = r.customer_id
group by c.first_name, c.last_name
having count(*) >= 40
order by count(*) desc;
```

```
first_name|last_name|rental_count|
----------+---------+------------+
ELEANOR   |HUNT     |          46|
KARL      |SEAL     |          45|
CLARA     |SHAW     |          42|
MARCIA    |DEAN     |          42|
TAMMY     |SANDERS  |          41|
SUE       |PETERS   |          40|
WESLEY    |BULL     |          40|
```

- count(*): 그룹화 한 전체 행의 수

3.7 Order by 절

- order by 절

```
ORDER BY [컬럼명 ¦ 컬럼번호] [ASC ¦ DESC]
```

  - 지정된 컬럼(열)을 기준으로 결과를 정렬 (다중 컬럼인 경우, 왼쪽부터 정렬)
  - 오름차순(ASC): 기본 정렬 값, 내림차순(DESC)

```
select c.first_name, c.last_name,
    time(r.rental_date) as rental_time
from customer as c inner join rental as r
    on c.customer_id = r.customer_id
where date(r.rental_date) = '2005-06-14'
order by c.last_name, c.first_name asc;
```

- 영화 대여 고객의 last_name을 기준으로 정렬
  - 오름 차순 정렬 (asc는 생략 가능)
- 고객 중 last_name이 동일한 경우,
  - first_name 으로 다시 정렬

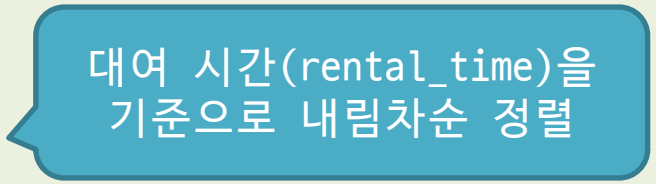
```
first_name¦last_name¦rental_time¦
----------+---------+-----------+
DANIEL    ¦CABRAL   ¦   23:09:38¦
CATHERINE ¦CAMPBELL ¦   23:17:03¦
HERMAN    ¦DEVORE   ¦   23:35:09¦
AMBER     ¦DIXON    ¦   23:42:56¦
JOYCE     ¦EDWARDS  ¦   23:16:26¦
JEANETTE  ¦GREENE   ¦   23:54:46¦
SONIA     ¦GREGORY  ¦   23:50:11¦
TERRENCE  ¦GUNDERSON¦   23:47:35¦
CHARLES   ¦KOWALSKI ¦   23:54:34¦
MATTHEW   ¦MAHAN    ¦   23:25:58¦
GWENDOLYN ¦MAY      ¦   23:16:27¦
MIRIAM    ¦MCKINNEY ¦   23:07:08¦
ELMER     ¦NOE      ¦   22:55:13¦
JEFFERY   ¦PINSON   ¦   22:53:33¦
MINNIE    ¦ROMERO   ¦   23:00:34¦
TERRANCE  ¦ROUSH    ¦   23:12:46¦
```

3.7 Order by 절

- 내림 차순 정렬: DESC

```
select c.first_name, c.last_name,
    time(r.rental_date) as rental_time
from customer as c
    inner join rental as r
    on c.customer_id = r.customer_id
where date(r.rental_date) = '2005-06-14'
order by time(r.rental_date) desc;
```

```
first_name|last_name|rental_time|
----------+---------+-----------+
JEANETTE  |GREENE   |   23:54:46|
CHARLES   |KOWALSKI |   23:54:34|
SONIA     |GREGORY  |   23:50:11|
TERRENCE  |GUNDERSON|   23:47:35|
AMBER     |DIXON    |   23:42:56|
HERMAN    |DEVORE   |   23:35:09|
MATTHEW   |MAHAN    |   23:25:58|
   .   .    .
TERRANCE  |ROUSH    |   23:12:46|
DANIEL    |CABRAL   |   23:09:38|
MIRIAM    |MCKINNEY |   23:07:08|
MINNIE    |ROMERO   |   23:00:34|
ELMER     |NOE      |   22:55:13|
JEFFERY   |PINSON   |   22:53:33|
```



3.7 Order by 절

- 순서를 통한 정렬
  - order by 다음에 정렬 기준이 되는 컬럼의 순서(index)를 사용

```
select c.first_name, c.last_name,
    time(r.rental_date) as rental_time
from customer as c
    inner join rental as r
    on c.customer_id = r.customer_id
where date(r.rental_date) = '2005-06-14'
order by 1 desc;
```

```
first_name|last_name|rental_time|
----------+---------+-----------+
TERRENCE  |GUNDERSON|   23:47:35|
TERRANCE  |ROUSH    |   23:12:46|
SONIA     |GREGORY  |   23:50:11|
MIRIAM    |MCKINNEY |   23:07:08|
MINNIE    |ROMERO   |   23:00:34|
MATTHEW   |MAHAN    |   23:25:58|
JOYCE     |EDWARDS  |   23:16:26|
   .   .   .
GWENDOLYN |MAY      |   23:16:27|
ELMER     |NOE      |   22:55:13|
DANIEL    |CABRAL   |   23:09:38|
CHARLES   |KOWALSKI |   23:54:34|
CATHERINE |CAMPBELL |   23:17:03|
AMBER     |DIXON    |   23:42:56|
```

first_name 컬럼의 index(1)를 기준으로 내림차순 정렬

3.8 연습 문제

- 실습 3-1
  - actor 테이블에서 모든 배우의 actor_id, first_name, last_name을 검색하고 last_name, first_name을 기준으로 오름 차순 정렬

```
select actor_id, first_name, last_name
from actor
order by last_name, first_name;
```

```
actor_id|first_name |last_name   |
--------+----------+------------+
      58|CHRISTIAN  |AKROYD      |
     182|DEBBIE     |AKROYD      |
      92|KIRSTEN    |AKROYD      |
     118|CUBA       |ALLEN       |
     145|KIM        |ALLEN       |
     194|MERYL      |ALLEN       |
      76|ANGELINA   |ASTAIRE     |
     112|RUSSELL    |BACALL      |
     190|AUDREY     |BAILEY      |
      67|JESSICA    |BAILEY      |
     115|HARRISON   |BALE        |
     187|RENEE      |BALL        |
      47|JULIA      |BARRYMORE   |
     158|VIVIEN     |BASINGER    |
     174|MICHAEL    |BENING      |
     124|SCARLETT   |BENING      |
      14|VIVIEN     |BERGEN      |
```

last_name이 동일한 경우, first_name 순으로 정렬

3.8 연습 문제

- 실습 3-2
  - 성이 'WILLIAMS' 또는 'DAVIS'인 모든 배우의 actor_id, first_name, last_name을 검색

```
select actor_id, first_name, last_name
from actor
where last_name = 'WILLIAMS' or last_name ='DAVIS';
```

```
actor_id¦first_name¦last_name¦
--------+----------+---------+
       4¦JENNIFER  ¦DAVIS    ¦
     101¦SUSAN     ¦DAVIS    ¦
     110¦SUSAN     ¦DAVIS    ¦
      72¦SEAN      ¦WILLIAMS ¦
     137¦MORGAN    ¦WILLIAMS ¦
     172¦GROUCHO   ¦WILLIAMS ¦
```

3.8 연습 문제

- 실습 3-3
  - rental 테이블에서 2005년 7월 5일 영화를 대여한 고객 ID를 반환하는 쿼리를 작성하고, date()함수로 시간 요소를 무시

```
select distinct customer_id
from rental
where date(rental_date) = '2005-07-05';
```

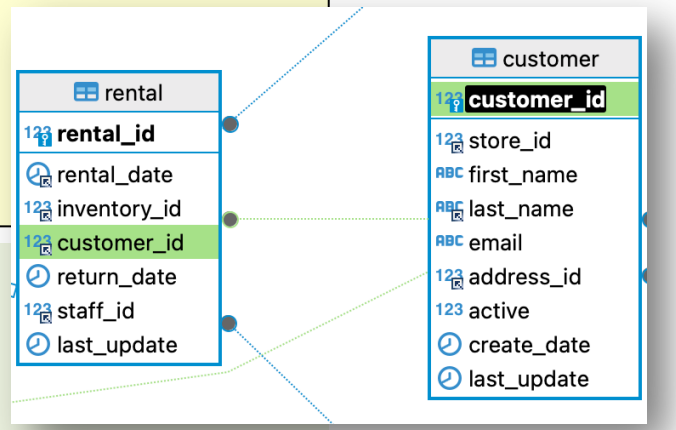
```
customer_id¦
-----------+
        565¦
        242¦
         37¦
         60¦
        594¦
          8¦
    . . .

        348¦
        553¦
        295¦
```

3.8 연습 문제

- 실습 3-4
  - 다음 결과를 참고하여 다중 테이블의 쿼리를 채우세요.

```
select c.store_id, c.email, r.rental_date, r.return_date
from customer as c inner join rental as r
    on c.customer_id  = r.customer_id
where date(r.rental_date) = '2005-06-14'
order by return_date desc;
```




```
store_id|email                                    |rental_date        |return_date        |
--------+-----------------------------------------+-------------------+-------------------+
       2|DANIEL.CABRAL@sakilacustomer.org         |2005-06-14 23:09:38|2005-06-23 22:00:38|
       1|TERRANCE.ROUSH@sakilacustomer.org        |2005-06-14 23:12:46|2005-06-23 21:53:46|
       1|MIRIAM.MCKINNEY@sakilacustomer.org       |2005-06-14 23:07:08|2005-06-21 17:12:08|
       1|GWENDOLYN.MAY@sakilacustomer.org         |2005-06-14 23:16:27|2005-06-20 02:40:27|
       1|JEANETTE.GREENE@sakilacustomer.org       |2005-06-14 23:54:46|2005-06-19 23:26:46|
       1|HERMAN.DEVORE@sakilacustomer.org         |2005-06-14 23:35:09|2005-06-19 03:20:09|
       2|JEFFERY.PINSON@sakilacustomer.org        |2005-06-14 22:53:33|2005-06-18 21:37:33|
       2|MATTHEW.MAHAN@sakilacustomer.org         |2005-06-14 23:25:58|2005-06-18 05:18:58|
       2|MINNIE.ROMERO@sakilacustomer.org         |2005-06-14 23:00:34|2005-06-18 01:58:34|
       1|SONIA.GREGORY@sakilacustomer.org         |2005-06-14 23:50:11|2005-06-17 21:44:11|
       1|TERRENCE.GUNDERSON@sakilacustomer.org    |2005-06-14 23:47:35|2005-06-17 05:28:35|
       2|ELMER.NOE@sakilacustomer.org             |2005-06-14 22:55:13|2005-06-17 02:11:13|
       2|JOYCE.EDWARDS@sakilacustomer.org         |2005-06-14 23:16:26|2005-06-16 21:00:26|
       1|AMBER.DIXON@sakilacustomer.org           |2005-06-14 23:42:56|2005-06-16 04:02:56|
       1|CHARLES.KOWALSKI@sakilacustomer.org      |2005-06-14 23:54:34|2005-06-16 02:26:34|
       2|CATHERINE.CAMPBELL@sakilacustomer.org    |2005-06-14 23:17:03|2005-06-15 20:43:03|
```

Questions?