The UPDATE statement

- Use this statement to update one or more columns
  - for all rows
  - for one or more rows matching WHERE clause conditions

To practice, first create a copy of a table

Create a copy of the film table, with data.

This does not copy the table's indexes or foreign keys

```
create table copy_film as select * from film;
```

Update all rows

Update the **release_year** to 2007 for all rows. Notice there is no WHERE clause.

```
update copy_film
set release_year = 2007;
```

Update all rows

Increase all rental rates by $1.00

```
update copy_film
set rental_rate = rental_rate + 1;
```

Undo the above update

```
update copy_film
set rental_rate = rental_rate – 1;
```

Update a filtered set of rows

Set the rental rate for all movies rated 'PG' to 0.99.

Step 1: first select all rows you want to update.

```
select * from copy_film
where rating = 'PG';
```

Update a filtered set of rows

Step 2: perform the update USING THE SAME WHERE CLAUSE

```
update copy_film
set rental_rate = 0.99
where rating = 'PG';
```

Update a filtered set of rows

Step 3: confirm the update

```
select rental_rate from copy_film
where rating = 'PG';
```

SAFETY FIRST!

Always perform a SELECT of the rows you want to update, and examine them before performing the update.

Updates without a WHERE clause will update all rows!

Update a filtered set of rows

Double the rental duration for films longer than 120 minutes.

```
select * from copy_film
where length > 120;
```

Update a filtered set of rows

Check the values before performing the update.

* in the expression `rental_duration * 2` means multiplication.

```
select film_id,
     rental_duration,
     rental_duration * 2 from copy_film
where length > 120;
```

Update a filtered set of rows

Perform the update using the same calculation and WHERE clause.

```
update copy_film
set rental_duration = rental_duration * 2
where length > 120;
```

Confirm the update

```
select film_id, rental_duration from copy_film
where length > 120;
```

Update multiple columns in a single statement

Double the rental duration **and** lower the rental rate by 50 cents for all movies rated G.

```
select film_id,
    rental_duration,
    rental_rate from copy_film
where rating = 'G';
```

Update multiple columns in a single statement

Perform the update

```
update copy_film
set rental_duration = rental_duration * 2,
    rental_rate = rental_rate - 0.5
where rating = 'G';
```

Updates using subqueries

Update all G-rated action films to PG-13

Step 1: Get all films in the 'Action' category

```
select film_id from film_category
inner join category
     on film_category.category_id = category.category_id
where category.name = 'Action';
```

Updates using subqueries

Update all G-rated action films to PG-13

Step 1: Get all films in the 'Action' category

```
select film_id from film_category
inner join category
     on film_category.category_id = category.category_id
where category.name = 'Action';
```

Updates using subqueries

Step 2: Get all films in the 'Action' category rated 'G'

```
select title, rating from copy_film
where film_id in (
      select film_id from film_category
      inner join category
      on film_category.category_id = category.category_id
      where category.name = 'Action'
)
and rating = 'G';
```

Updates using subqueries

Step 3: Perform the update using the same WHERE clause

```
update copy_film
set rating = 'PG-13'
where film_id in (
      select film_id from film_category
      inner join category
      on film_category.category_id = category.category_id
      where category.name = 'Action'
)
and rating = 'G';
```

Updates using JOINs

Load the world database from the "MySQL samples" folder in Moodle,
or from https://dev.mysql.com/doc/index-other.html (world.sql.zip)

Updates using JOINs

Examine the data

```
use world;

select * from city;
select * from country;
select * from countrylanguage;
```

Updates using JOINs

Append the country name to the city name field. For purposes of this example, we first increase the allowable maximum length of a city name from 35 to 100.

Note the use of the ALTER TABLE statement.

```
describe city;
alter table city modify Name varchar(100);
```

Updates using JOINs

Append country name to city name, seperated by a comma.

```
update city, country
set city.Name = concat(city.Name, ', ', country.Name)
where city.CountryCode = country.Code;
-- join condition
```

Updates using JOINs

Confirm the changes

```
select * from city;
```

Updates using JOINs

Undo the previous update by using the substring_index() function. This function is used in this example to get the part of the city name before the comma.

```
select city.Name,
     substring_index(city.Name, ',', 1)
from city;

update city
set city.Name = substring_index(city.Name, ',', 1);
```

Updates using JOINs

Confirm the changes

```
select * from city;
```

Undo the **city.Name** column definition change

```
alter table city modify Name varchar(35);
```

DELETE

- Delete all rows
- Delete rows matching WHERE clause
- Delete rows using subquery
- ON DELETE CASCADE: https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html