Lab2 SQL Intro

Lab2 Agenda

- SQL Overview
- Command-Line Tools
- Query Primer / Select
- Εργαστηριακές Ασκήσεις
- Εξαμηνιαία Εργασία

Structured Query Language (SQL)

```
UPDATE clause  {UPDATE country                                   }
                                        expression               }
   SET clause  {SET population = population + 1                  } statement
                                 expression                      }
 WHERE clause  {WHERE name = 'USA' ;                              }
                       predicate                                  }
```

- domain-specific language (DSL) language
- managing data held in a relational database management system (RDBMS)
- declarative language (4GL) - with procedural elements
- ANSI - ISO standard
- Newer versions support arrays, XML, JSON

DDL & DML

Data Definition Language (DDL)

create the structure of the database and the other database objects

1. **create** : create new databases and tables
2. **drop** : delete databases and tables
3. **alter** : modify an already existing database object such as a table

Data Manipulation Language (DML)

manage the data stored in the database

1. **insert** : store new records or rows to the table
2. **update** : modify an existing record in the table
3. **delete** : deleting a certain record or a set of records from the table
4. **select** : retrieving specific records from one or more tables - DQL: Data Query Language

Data Control Language (DCL)

rights, permissions and other controls of the database system

1. **GRANT** : allow users access privileges to database
2. **REVOKE** : withdraw users access privileges given by using the GRANT command

Transaction Control Language (TCL)

rights, permissions and other controls of the database system

1. **COMMIT** : commits a transaction
2. **ROLLBACK** : rollback a transaction in case of any error occurs
3. **SAVEPOINT** : a point inside a transaction that allows rollback state to what it was at the time of the savepoint
4. **SET TRANSACTION** : specify characteristics for the transaction

First Steps

- Σύνδεση

```
mysql -u root -p
```

```
# mysql -u root -p
Enter password: ****************
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.22-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

show databases

- Εμφάνιση προσβάσιμων βάσεων δεδομένων

```
show databases;
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| chinook            |
| dbprojectdemo      |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sakila             |
   ...
| world              |
+--------------------+
11 rows in set (0.024 sec)
```

USE database

- Χρήση συγκεκριμένης βάσης δεδομένων

```
USE sakila;
```

- Έξοδος από τον mysql client

```
QUIT (ή EXIT);
```

Σύνδεση με επιλογή ΒΔ

```
mysql -u root -p sakila
```

Exploring: see the tables available

```
MariaDB [sakila]> show tables;
+----------------------------+
| Tables_in_sakila           |
+----------------------------+
| actor                      |
| actor_info                 |
| address                    |
| category                   |
| city                       |
| country                    |
| customer                   |
| customer_list              |
| film                       |
| film_actor                 |
        ...
| staff                      |
| staff_list                 |
| store                      |
+----------------------------+
23 rows in set (0.001 sec)
```

The Sakila Database

Table Name	Definition
film	A movie that has been released and can be rented
actor	A person who acts in films
customer	A person who watches films
category	A genre of films
payment	A rental of a film by a customer
language	A language spoken by the actors of a film
film_actor	An actor in a film
inventory	A film available for rental

Exploring: look at the columns in a table

```
MariaDB [sakila]> desc actor;
+-------------+------------------+------+-----+---------------------+-------------------------------+
| Field       | Type             | Null | Key | Default             | Extra                         |
+-------------+------------------+------+-----+---------------------+-------------------------------+
| actor_id    | int(10) unsigned | NO   | PRI | NULL                | auto_increment                |
| first_name  | varchar(45)      | NO   |     | NULL                |                               |
| last_name   | varchar(45)      | NO   | MUL | NULL                |                               |
| last_update | timestamp        | NO   |     | current_timestamp() | on update current_timestamp() |
+-------------+------------------+------+-----+---------------------+-------------------------------+
4 rows in set (0.025 sec)
```

Command-Line Tool PostgreSQL (psql)

- \q to exit
- chcp: set encoding to windows-1253

```
D:\dev_tools\PostgreSQL\bin>chcp 1253
Active code page: 1253

D:\dev_tools\PostgreSQL\bin>psql.exe user=postgres
Κωδικός πρόσβασης για τον χρήστη postgres:
psql (14.2)
Γράψτε «help» για βοήθεια.
postgres=#
```

PostgreSQL (psql) Exploring: list databases

```
postgres=# \l
                                      Λίστα βάσεων δεδομένων
   Όνομα   | Ιδιοκτήτης | Κωδικοποίηση |      Σύνθεση      |       Ctype       |  Προνόμια πρόσβασης
-----------+------------+--------------+-------------------+-------------------+-----------------------
 postgres  | postgres   | UTF8         | Greek_Greece.1253 | Greek_Greece.1253 |
 sakila    | postgres   | UTF8         | Greek_Greece.1253 | Greek_Greece.1253 |
 template0 | postgres   | UTF8         | Greek_Greece.1253 | Greek_Greece.1253 | =c/postgres          +
           |            |              |                   |                   | postgres=CTc/postgres
 template1 | postgres   | UTF8         | Greek_Greece.1253 | Greek_Greece.1253 | =c/postgres          +
           |            |              |                   |                   | postgres=CTc/postgres
 testdb    | xdataadmin | UTF8         | Greek_Greece.1253 | Greek_Greece.1253 |
 xdata     | xdataadmin | UTF8         | Greek_Greece.1253 | Greek_Greece.1253 |
(6 σειρές)
```

PostgreSQL (psql) Exploring: switching Databases

```
postgres-# \c sakila
Τώρα είστε συνδεδεμένοι στη βάση δεδομένων ?sakila? ως χρήστης ?postgres?.
sakila-#
```

PostgreSQL (psql) Exploring: Describe

```
sakila=# \d
                        Λίστα σχέσεων
 Σχήμα  |            Όνομα             |   Τύπος    | Ιδιοκτήτης
--------+-----------------------------+------------+-------------
 public | actor                       | πίνακας    | postgres
 public | actor_actor_id_seq          | ακολουθία  | postgres
 public | actor_info                  | όψη        | postgres
 public | address                     | πίνακας    | postgres
 public | address_address_id_seq      | ακολουθία  | postgres
 public | category                    | πίνακας    | postgres
 public | category_category_id_seq    | ακολουθία  | postgres
```

PostgreSQL (psql) Exploring: listing Tables

```
sakila-# \dt
                  Λίστα σχέσεων
 Σχήμα  |      Όνομα       |  Τύπος  | Ιδιοκτήτης
--------+------------------+---------+-------------
 public | actor            | πίνακας | postgres
 public | address          | πίνακας | postgres
 public | category         | πίνακας | postgres
                 ...
 public | staff            | πίνακας | postgres
 public | store            | πίνακας | postgres
(21 σειρές)
```

PostgreSQL (psql) Exploring: Describe Table

```
sakila=# \d actor
                                        Πίνακας  «public.actor»
    Στήλη    |            Τύπος            | Σύνθεση | Nullable |             Προκαθορισμένο
-------------+-----------------------------+---------+----------+-------------------------------------
 actor_id    | integer                     |         | not null | nextval('actor_actor_id_seq'::regclass)
 first_name  | character varying(45)       |         | not null |
 last_name   | character varying(45)       |         | not null |
 last_update | timestamp without time zone |         | not null | now()
Ευρετήρια:
    "actor_pkey" PRIMARY KEY, btree (actor_id)
    "idx_actor_last_name" btree (last_name)
Αναφέρεται από:
    TABLE "film_actor" CONSTRAINT "film_actor_actor_id_fkey" FOREIGN KEY (actor_id) REFERENCES actor(actor_id) ON UPDATE CASCADE ON DELETE RESTRICT
Triggers:
    last_updated BEFORE UPDATE ON actor FOR EACH ROW EXECUTE FUNCTION last_updated()
```

PostgreSQL (psql) Exploring: Query

```
sakila=# select * from actor limit 1;
 actor_id | first_name | last_name |     last_update
----------+------------+-----------+---------------------
        1 | PENELOPE   | GUINESS   | 2006-02-15 04:34:33
(1 σειρά)
```

Data Types

1. Character Data

```
char(20) /* fixed-length  limit: 255 bytes*/
varchar(20) /* variable-length limit: 65,535 bytes*/
```

- tinytext, text, mediumtext, longtext

Data Types

2. Numeric Data

```
int(10)
float(4,2) /* total of four digits, two to the left of the decimal and two to the right of the decimal*/
```

- **integer** types: tinyint, smallint, mediumint, int, bigint
- **floating-point** types: float(p, s), double(p,s)
- precision (p) the total number of digits allowed in this column
- scale (s) the number of decimal places to the right (if positive) {or left (if negative; this is rarely used)} of the decimal point.

Data Types

3. Temporal Data 📅

Type	Default format
date	YYYY-MM-DD
datetime	YYYY-MM-DD HH:MI:SS
timestamp	YYYY-MM-DD HH:MI:SS
year	YYYY
time	HH:MI:SS

- 📝 database servers
  - store temporal data in various ways
  - offer different range of dates for temporal columns

- mysql

```
MariaDB [sakila]> select now(), current_date(),current_time();
+---------------------+----------------+----------------+
| now()               | current_date() | current_time() |
+---------------------+----------------+----------------+
| 2022-03-15 14:12:48 | 2022-03-15     | 14:12:48       |
+---------------------+----------------+----------------+
1 row in set (0.002 sec)
```

- PostgreSQL

```
select now(), CURRENT_TIME, CURRENT_TIMESTAMP;
```

Query Primer

- Each time a query is sent to the server, the server checks the following things prior to statement execution:
  - Do you have permission to execute the statement? 🆔
  - Do you have permission to access the desired data? 🗹
  - Is your statement syntax correct? 📒

1. query optimizer 🔁
2. execution plan 🕛

Select Query Clause

Clause	Purpose
select	Determines which columns to include in the query's result set
from	Identifies the tables from which to retrieve data and how the tables should be joined
where	Filters out unwanted data
group by	Used to group rows together by common column values
having	Filters out unwanted groups
order by	Sorts the rows of the final result set by one or more columns

Select

1. ⭐ *

```
SELECT * FROM language;
/* When you see SELECT *, think of it like asking your SQL software to
SELECT ALL THE COLUMNS*/
/* Show me all the columns and all the rows in the language table*/
```

```
MariaDB [sakila]> SELECT * 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 |
+-------------+----------+---------------------+
6 rows in set (0.069 sec)
```

or specific fields

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

```
MariaDB [sakila]> 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 |
+-------------+----------+---------------------+
6 rows in set (0.000 sec)
```

you can include:

- Literals, such as numbers or strings
- Expressions, such as transaction.amount * −1
- Built-in function calls, such as ROUND(transaction.amount, 2)
- User-defined function calls

Column Aliases

assign your own labels

```
SELECT language_id, 'COMMON' language_usage,
       language_id * 3.1415927 lang_pi_value, upper(name) language_name
FROM language;
```

```
MariaDB [sakila]> SELECT language_id, 'COMMON' language_usage,
    ->        language_id * 3.1415927 lang_pi_value, upper(name) language_name
    -> FROM language;
+--------------+----------------+---------------+----------------+
| language_id | language_usage | lang_pi_value | language_name |
+--------------+----------------+---------------+----------------+
|           1 | COMMON         |     3.1415927 | ENGLISH        |
|           2 | COMMON         |     6.2831854 | ITALIAN        |
|           3 | COMMON         |     9.4247781 | JAPANESE       |
|           4 | COMMON         |    12.5663708 | MANDARIN       |
|           5 | COMMON         |    15.7079635 | FRENCH         |
|           6 | COMMON         |    18.8495562 | GERMAN         |
+--------------+----------------+---------------+----------------+
6 rows in set (0.020 sec)
```

Removing Duplicates

```
SELECT actor_id FROM film_actor ORDER BY actor_id;
```

```
MariaDB [sakila]> SELECT actor_id FROM film_actor ORDER BY actor_id;
+----------+
| actor_id |
+----------+
|        1 |
|        1 |
    ...
|      200 |
|      200 |
+----------+
5462 rows in set (0.025 sec)
```

vs

```
SELECT DISTINCT actor_id FROM film_actor ORDER BY actor_id;
/* distinct set of results requires the data to be sorted */
```

```
MariaDB [sakila]> SELECT DISTINCT actor_id FROM film_actor ORDER BY actor_id;
+----------+
| actor_id |
+----------+
|        1 |
|        2 |
|        3 |
    ...
|      200 |
+----------+
200 rows in set (0.001 sec)
```

from

- The from clause defines the tables used by a query, along with the means of linking the **tables** together
  1. **Permanent** tables (i.e., created using the create table statement)
  2. **Derived** tables (i.e., rows returned by a subquery and held in memory)
  3. **Temporary** tables (i.e., volatile data held in memory)
  4. **Virtual** tables (i.e., created using the create view statement)

2. Derived (subquery-generated) tables

- subquery: query contained within another query

```
SELECT concat(cust.last_name, ', ', cust.first_name) full_name
FROM
 ( SELECT first_name, last_name, email
   FROM customer
   WHERE first_name = 'JESSIE'
 ) cust;
```

```
MariaDB [sakila]> SELECT concat(cust.last_name, ', ', cust.first_name) full_name
    -> FROM
    ->  ( SELECT first_name, last_name, email
    ->    FROM customer
    ->    WHERE first_name = 'JESSIE'
    ->  ) cust;
+----------------+
| full_name      |
+----------------+
| BANKS, JESSIE |
| MILAM, JESSIE |
+----------------+
2 rows in set (0.025 sec)
```

3. Temporary tables

```
CREATE TEMPORARY TABLE actors_j
(actor_id smallint(5),  first_name varchar(45),  last_name varchar(45) );
desc actors_j;
```

```
MariaDB [sakila]> CREATE TEMPORARY TABLE actors_j
    -> (actor_id smallint(5),  first_name varchar(45),  last_name varchar(45) );
Query OK, 0 rows affected (0.016 sec)

MariaDB [sakila]> desc actors_j;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| actor_id   | smallint(5) | YES  |     | NULL    |       |
| first_name | varchar(45) | YES  |     | NULL    |       |
| last_name  | varchar(45) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.019 sec)
```

4. Views

- view: a query that is stored in the data dictionary.
- It looks and acts like a table but without any data (~**virtual table**)

```
CREATE VIEW cust_vw AS
  SELECT customer_id, first_name, last_name, active
  FROM customer;
desc cust_vw;
```

```
MariaDB [sakila]> CREATE VIEW cust_vw AS
    ->   SELECT customer_id, first_name, last_name, active
    ->   FROM customer;
Query OK, 0 rows affected (0.026 sec)

MariaDB [sakila]> desc cust_vw;
+-------------+------------------+------+-----+---------+-------+
| Field       | Type             | Null | Key | Default | Extra |
+-------------+------------------+------+-----+---------+-------+
| customer_id | int(10) unsigned | NO   |     | 0       |       |
| first_name  | varchar(45)      | NO   |     | NULL    |       |
| last_name   | varchar(45)      | NO   |     | NULL    |       |
| active      | tinyint(1)       | NO   |     | 1       |       |
+-------------+------------------+------+-----+---------+-------+
4 rows in set (0.013 sec)
```

```sql
SELECT first_name, last_name
   FROM cust_vw
   WHERE active = 0;
```

```
MariaDB [sakila]> SELECT first_name, last_name
    ->    FROM cust_vw
    ->    WHERE active = 0;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| SANDRA     | MARTIN    |
| JUDITH     | COX       |
| HEIDI      | LARSON    |
             ...
| JIMMIE     | EGGLESTON |
| TERRANCE   | ROUSH     |
+------------+-----------+
15 rows in set (0.014 sec)
```

where

- way to filter out those rows that are not of interest

```
SELECT title
FROM film
WHERE rating = 'G' AND rental_duration >= 7;
/* all conditions must evaluate to true */
```

```
MariaDB [sakila]>  SELECT title
    ->  FROM film
    ->  WHERE rating = 'G' AND rental_duration >= 7;
+--------------------------+
| title                    |
+--------------------------+
| BLANKET BEVERLY          |
| BORROWERS BEDAZZLED      |
| BRIDE INTRIGUE           |
      ...
| WAKE JAWS                |
| WAR NOTTING              |
+--------------------------+
29 rows in set (0.037 sec)
```

```sql
SELECT title
FROM film
WHERE rating = 'G' OR rental_duration >= 7;
/* only one of the conditions needs to evaluate to true */
```

```
MariaDB [sakila]>  SELECT title  FROM film  WHERE rating = 'G' OR rental_duration >= 7;
+----------------------------+
| title                      |
+----------------------------+
| ACE GOLDFINGER             |
| ADAPTATION HOLES           |
| AFFAIR PREJUDICE           |
        ...
| WON DARES                  |
| WORKER TARZAN              |
| YOUNG LANGUAGE             |
+----------------------------+
340 rows in set (0.001 sec)
```

```sql
SELECT title, rating, rental_duration FROM film
   WHERE (rating = 'G' AND rental_duration >= 7)
   OR (rating = 'PG-13' AND rental_duration < 4);
/* use parentheses to separate groups of conditions when mixing different operators  */
```

```
MariaDB [sakila]> 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 |
                            ...
| WAR NOTTING             | G      |               7 |
| WORLD LEATHERNECKS      | PG-13  |               3 |
+-------------------------+--------+-----------------+
68 rows in set (0.001 sec)
```

group by / having

- *group by*: group rows together by common column values/ summarize unique combinations of columns values
- *having*: Filters out unwanted groups
  - select all of the customers who have rented 40 or more films ❓

```sql
SELECT c.first_name, c.last_name, count(*)
  FROM customer c
  INNER JOIN rental r
  ON c.customer_id = r.customer_id
  GROUP BY c.first_name, c.last_name
  HAVING count(*) >= 40;
```

```
MariaDB [sakila]> SELECT c.first_name, c.last_name, count(*)  FROM customer c
    ->   INNER JOIN rental r
    ->   ON c.customer_id = r.customer_id
    ->   GROUP BY c.first_name, c.last_name
    ->   HAVING count(*) >= 40;
+------------+-----------+----------+
| first_name | last_name | count(*) |
+------------+-----------+----------+
| CLARA      | SHAW      |       42 |
| ELEANOR    | HUNT      |       46 |
| KARL       | SEAL      |       45 |
| MARCIA     | DEAN      |       42 |
| SUE        | PETERS    |       40 |
| TAMMY      | SANDERS   |       41 |
| WESLEY     | BULL      |       40 |
+------------+-----------+----------+
7 rows in set (0.044 sec)
```

order by

- order by: Sorts the rows of the final result set by one or more columns

```
SELECT * FROM  actor order by last_name;
```

```
MariaDB [sakila]> select * from actor order by last_name;
+----------+-------------+--------------+---------------------+
| actor_id | first_name  | last_name    | last_update         |
+----------+-------------+--------------+---------------------+
|      182 | DEBBIE      | AKROYD       | 2006-02-15 04:34:33 |
|       92 | KIRSTEN     | AKROYD       | 2006-02-15 04:34:33 |
                        ...
|      186 | JULIA       | ZELLWEGER    | 2006-02-15 04:34:33 |
|      111 | CAMERON     | ZELLWEGER    | 2006-02-15 04:34:33 |
+----------+-------------+--------------+---------------------+
200 rows in set (0.001 sec)
```

```
MariaDB [sakila]> select * from actor order by last_name DESC;
+----------+-------------+-------------+---------------------+
| actor_id | first_name  | last_name   | last_update         |
+----------+-------------+-------------+---------------------+
|      186 | JULIA       | ZELLWEGER   | 2006-02-15 04:34:33 |
|       85 | MINNIE      | ZELLWEGER   | 2006-02-15 04:34:33 |
                            ...
|       92 | KIRSTEN     | AKROYD      | 2006-02-15 04:34:33 |
|       58 | CHRISTIAN   | AKROYD      | 2006-02-15 04:34:33 |
|      182 | DEBBIE      | AKROYD      | 2006-02-15 04:34:33 |
+----------+-------------+-------------+---------------------+
200 rows in set (0.001 sec)
```

Εργαστηριακές Ασκήσεις 💻

1. Select actor ID, first name, and last name for all actors (Sort by last name and then by first name)
2. Select the actor ID, first name, and last name for all actors whose last name equals 'WILLIAMS' or 'DAVIS'
3. Select the title, description, rating, and length columns for films that last 3 hours or longer;
4. Which actors have the last name 'Johansson';
5. How many distinct actors last names are there?
6. Which last names are not repeated?
7. Which last names appear more than once?
8. What is that average length of all the films?

Εξαμηνιαία Εργασία 💻

- Εκφώνηση🔗
- Database Schema Design
  1. Start thinking about the entities you need
     - Identify entities, attributes and relationships from the problem description
     - identify cardinality ratios of the relationships found
  2. Design an E/R diagram for your database
     - Look for any issues that are apparent in the E/R diagram

Wrap Up

1. [x] `SQL Overview`
2. [x] `Command-Line Tools`
3. [x] `Query Primer / Select`
4. [x] `Εργαστηριακές Ασκήσεις`
5. [x] `Εξαμηνιαία Εργασία`

Wrap Up

😱 **Απορίες** https://discord.gg/g3fFxWVPfD

Εργαστηριακές Ασκήσεις / Απαντήσεις

1. Select actor ID, first name, and last name for all actors (Sort by last name and then by first name)

```
SELECT actor_id, first_name, last_name
  FROM actor
  ORDER BY 3,2;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

2. Select the actor ID, first name, and last name for all actors whose last name equals 'WILLIAMS' or 'DAVIS'

```
SELECT actor_id, first_name, last_name
  FROM actor
  WHERE last_name IN ('WILLIAMS','DAVIS');
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

3. Select the title, description, rating, and length columns for films that last 3 hours or longer;

```
SELECT title, description, rating, length AS "movie length"
FROM film  WHERE length >= 180;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

4. Which actors have the last name 'Johansson';

```
SELECT * FROM actor WHERE last_name = 'Scarlett';
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

5. How many distinct actors last names are there?

```
SELECT count(DISTINCT last_name) FROM actor;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

6. Which last names are not repeated?

```
SELECT last_name FROM actor
GROUP BY last_name
HAVING count(*) = 1;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

7. Which last names appear more than once?

```
SELECT last_name FROM actor
GROUP BY last_name
HAVING count(*) > 1;
```

Εργαστηριακές Ασκήσεις / Απαντήσεις

8. What is that average length of all the films?

```
select avg(length) from film;
```