ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

Εργασία: Βάσεις Δεδομένων

Στοιχεία Φοιτητών:

Αλευράς Ηλίας 1069667 up1069667@upnet.gr

Στυλιανού Σάββας 1069661 up1069661 up1069661@upnet.gr

31 Αυγούστου 2022

ΚΕΦΑΛΑΙΟ 1 & 2: ΔΗΜΙΟΥΡΓΙΑ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Για το σχεδιασμό της αναθεωρημένης βάσης, προχωρήσαμε στη δημιουργία νέων πινάκων αλλά και στην τροποποίηση πινάκων που υπήρχαν ήδη στη βάση δεδομένων. Με δεδομένο ότι ο πάροχος του τηλεοπτικού περιεχομένου επιθυμεί να αναβαθμίσει την υπηρεσία και να δώσει τη δυνατότητα στους εγγεγραμμένους χρήστες να ενοικιάζουν και σειρές, εργαστήκαμε προς αυτή την κατεύθυνση για να επεκτείνουμε τη βάση δεδομένων (ΒΔ) και να προσθέσουμε τη δυνατότητα να διατηρούμε στη βάση επιπρόσθετες πληροφορίες.

Αρχικά, δημιουργήσαμε δύο πίνακες για τις τρεις κατηγορίες χρηστών που θα συνδέονται με τη βάση δεδομένων, τον πίνακα για τους υπαλλήλους **(employees)** και τους διαχειριστές **(administrator)** του συστήματος. Ο πίνακας πελάτες **(customer)** υπάρχει ήδη στη βάση δεδομένων μας. Στους παραπάνω πίνακες αποθηκεύουμε ένα μοναδικό κωδικό, το όνομα και το επίθετο, αλλά και το email κάθε υπαλλήλου και διαχειριστή. Με τη χρήση της εντολής **INSERT** προσθέτουμε στους πίνακες πέντε υπαλλήλους και τρεις διαχειριστές, αντίστοιχα.

TABLE employees

```
MariaDB [tvondemand]> SELECT * FROM employees LIMIT 5;
+-------------+------------+-------------+-----------------------------------------+
| employee_id | first_name | last_name   | email                                   |
+-------------+------------+-------------+-----------------------------------------+
|         123 | Alexander  | Green       | alex.green@sakilaemployee.org           |
|         124 | Athena     | Pappa       | athena.pappa@sakilaemployee.org         |
|         125 | Andreas    | Eleftherakis | andreas.eleftherakis@sakilaemployee.org |
|         126 | George     | Ioannou     | george.ioannou@sakilaemployee.org       |
|         127 | Periklis   | Smith       | periklis.smith@sakilaemployee.org       |
+-------------+------------+-------------+-----------------------------------------+
5 rows in set (0.000 sec)
```

TABLE administrator

```
MariaDB [tvondemand]> SELECT * FROM administrator LIMIT 5;
+------------------+------------+-----------+--------------------------------+
| administrator_id | first_name | last_name | email                          |
+------------------+------------+-----------+--------------------------------+
|              211 | Elena      | Vlachou   | elena.vlachou@sakiladmin.org   |
|              212 | Nikos      | Pappas    | nikos.pappas@sakiladmin.org    |
|              213 | Andreas    | Iacovou   | andreas.iacovou@sakiladmin.org |
+------------------+------------+-----------+--------------------------------+
3 rows in set (0.000 sec)
```

Όπως αναφέρεται στην περιγραφή της άσκησης, απατείται επίσης η προσθήκη ενός πίνακα για τη διατήρηση των ενεργειών που εκτελούνται από οποιοδήποτε χρήστη σε συγκεκριμένους πίνακες στη βάση δεδομένων, ο οποίος θα χρησιμοποιηθεί στην άσκηση του Μέρους B.

Στον πίνακα διατήρησης ενεργειών **(log_table)**, διατηρούμε συγκεκριμένες πληροφορίες όπως το username του χρήστη, την κατηγορία του χρήστη (πελάτης, υπάλληλος, διαχειριστής), την ενέργεια που εκτέλεσε (εισαγωγή, επεξεργασία, διαγραφή), τον πίνακα που αφορούσε η ενέργεια, την ημερομηνία και ώρα της ενέργειας και αν αυτή εκτελέστηκε επιτυχώς ή όχι.
Πιο κάτω, παραθέτουμε τη μορφή του πίνακα ενεργειών και ένα παράδειγμα με την προσθήκη μίας ενέργειας από ένα διαχειριστή στον πίνακα rental.

TABLE log_table

```
MariaDB [tvondemand]> select * from log_table;
+------------------------------+---------------+-------------+--------------+---------------------+---------------+
| username                     | user_type     | action_type | action_table | action_datetime_    | if_successful |
+------------------------------+---------------+-------------+--------------+---------------------+---------------+
| elena.vlachou@sakiladmin.org | Administrator | update      | RENTAL       | 2022-08-29 12:00:00 |             1 |
+------------------------------+---------------+-------------+--------------+---------------------+---------------+
1 row in set (0.000 sec)
```

Επιστρέφοντας στο σχεδιασμό της βάσης δεδομένων και στις αλλαγές που έχουμε διεκπεραιώσει, θα αναφερθούμε σε αυτές παρακάτω.

Αρχικά, δημιουργούμε τον πίνακα **series**, στον οποίο διατηρούμε όλες τις πληροφορίες για τις διαθέσιμες σειρές. Στον πίνακα αυτό υπάρχουν πληροφορίες ανάλογες με τον πίνακα ταινιών, αλλά και επιπλέον στοιχεία όπως οι κύκλοι της σειράς. Στον πίνακα αυτό το **series_id** αποτελεί το πρωτεύων κλειδί του πίνακα, καθώς κάθε σειρά υπάρχει ένας μοναδικός κωδικός. Στον πίνακα αυτό υπάρχουν και οι απαραίτητοι περιορισμοί και συσχετισμοί με τον πίνακα **language** για τις στήλες language_id και original_language_id.
Με τη χρήση της εντολής **INSERT** προσθέτουμε πληροφορίες για επτά σειρές.
TABLE series

```
MariaDB [tvondemand]> SELECT * FROM series LIMIT 5;
| series_id | title            | description                                                                                                    | num_seasons | language_id | origi
nal_language_id | rating | special_features |
|      1001 | The Sandman      | Upon escaping after decades of imprisonment by a mortal wizard, Dream, the personification of dreams, sets about to reclaim his lost equipment |           1 |           1 |
           NULL | G      | Deleted Scenes   |
|      1002 | Better Call Saul | The trials and tribulations of criminal lawyer Jimmy McGill before his fateful run-in with Walter White and Jesse Pinkman |           6 |           1 |
              1 | NULL   | NULL             |
|      1003 | Stranger Things  | When a young boy disappears, his mother, a police chief and his friends must confront terrifying supernatural forces in order to get him back |           4 |           1 |
              1 | G      | Behind the Scenes |
|      1004 | Game of Thrones  | Nine noble families fight for control over the lands of Westeros, while an ancient enemy returns after being dormant for millennia |           8 |           1 |
           NULL | G      | NULL             |
|      1005 | The Boys         | A group of vigilantes set out to take down corrupt superheroes who abuse their superpowers |           3 |           1 |
              1 | G      | Commentaries     |
5 rows in set (0.000 sec)
```

Στη συνέχεια, δημιουργούμε τον πίνακα **series_seasons**, στον οποίο αποθηκεύουμε πληροφορίες σχετικά με τους κύκλους κάθε σειράς. Στον πίνακα υπάρχει ο μοναδικός αριθμός της σειράς ο οποίος συνδέεται ως **FOREIGN KEY** με τον πίνακα **series**, ο αριθμός του κύκλου, το έτος κυκλοφορίας κάθε κύκλου και το σύνολο των επεισοδίων.

Με τη χρήση της εντολής **INSERT** προσθέτουμε πληροφορίες για 31 κύκλους σειρών.

TABLE series_seasons

```
MariaDB [tvondemand]> SELECT * FROM series_seasons LIMIT 5;
+-----------+-----------+---------------------+--------------+
| series_id | season_id | season_release_year | num_episodes |
+-----------+-----------+---------------------+--------------+
|      1001 |         1 |                2022 |           11 |
|      1002 |         1 |                2015 |           10 |
|      1002 |         2 |                2016 |           10 |
|      1002 |         3 |                2017 |           10 |
|      1002 |         4 |                2018 |           10 |
+-----------+-----------+---------------------+--------------+
5 rows in set (0.000 sec)
```

Για την ομαλή διατήρηση των πληροφοριών σχετικά με τις σειρές, δημιουργούμε ένα ακόμα πίνακα, τον πίνακα **episodes** στον οποίο έχουμε ως πρωτεύων κλειδί το episode_id το οποίο είναι μοναδικό. Διατηρούμε επίσης το series_id ως foreign key από τον πίνακα series, αλλά και τον μοναδικό αριθμό του κύκλου της σειράς που περιλαμβάνει κάθε επεισόδιο. Με τη χρήση της εντολής **INSERT** προσθέτουμε 303 επεισόδια που είναι πλέον διαθέσιμα για ενοικίαση.

TABLE episodes

```
MariaDB [tvondemand]> SELECT * FROM episodes LIMIT 5;
+------------+-----------+-----------+
| episode_id | series_id | season_id |
+------------+-----------+-----------+
|        111 |      1001 |         1 |
|        112 |      1001 |         1 |
|        113 |      1001 |         1 |
|        114 |      1001 |         1 |
|        115 |      1001 |         1 |
+------------+-----------+-----------+
5 rows in set (0.000 sec)
```

Ο πίνακας αυτός είναι αρκετά σημαντικός, καθώς σε αυτόν αποθηκεύουμε τις πληροφορίες για κάθε επεισόδιο μιας σειράς. Λαμβάνοντας υπόψη πως η εταιρεία χρεώνει κάθε πελάτη ανά επεισόδιο σειράς που βλέπει, θεωρούμε πως αυτός ο πίνακας είναι αντίστοιχος με τον πίνακα film. Κι αυτό γιατί τόσο για την εταιρεία, όσο και για τον πελάτη, μία ταινία και ένα επεισόδιο σειράς είναι ένα αντικείμενο που τίθεται διαθέσιμο για ενοικίαση.

Συνεχίζουμε το σχεδιασμό της βάσης δημιουργώντας νέους πίνακες, τροποποιώντας ή επανασχεδιάζοντας τους υφιστάμενους πίνακες με σκοπό να δημιουργήσουμε μια λειτουργική βάση δεδομένων τόσο για την εταιρεία, τους διαχειριστές και τους υπαλλήλους, αλλά και για τους πελάτες.
Δημιουργούμε τον πίνακα **series_actors** όπου διατηρούμε πληροφορίες για τους ηθοποιούς κάθε σειράς. Στον πίνακα διατηρούμε τον μοναδικό αριθμό για κάθε

ηθοποιό και τη σειρά στην οποία συμμετέχει. Υπάρχει η σύνδεση με τον πίνακα series για το series_id και με τον πίνακα actor για τον actor_id, ενώ με την εντολή **INSERT** προσθέτουμε τρεις ηθοποιούς για κάθε σειρά.

TABLE series_actor

```
MariaDB [tvondemand]> SELECT * FROM series_actor LIMIT 5;
+----------+-----------+
| actor_id | series_id |
+----------+-----------+
|    10011 |      1001 |
|    10012 |      1001 |
|    10013 |      1001 |
|    10014 |      1002 |
|    10015 |      1002 |
+----------+-----------+
5 rows in set (0.000 sec)
```

Προσθέτουμε επίσης τον πίνακα **series_category** με τον οποίο διατηρούμε πληροφορίες για την κατηγορία κάθε σειράς. Χρησιμοποιούμε τα series_id και category_id ως foreign keys από τους πίνακες series και category αντίστοιχα.

TABLE series_category

```
MariaDB [tvondemand]> select * from series_category;
+-----------+-------------+
| series_id | category_id |
+-----------+-------------+
|      1001 |          11 |
|      1002 |           7 |
|      1003 |           7 |
|      1004 |           1 |
|      1005 |           1 |
|      1006 |          14 |
|      1007 |           7 |
+-----------+-------------+
7 rows in set (0.047 sec)
```

Τροποποιούμε τον πίνακα **inventory** που λειτουργεί ως κατάλογος για τις διαθέσιμες ταινίες και επεισόδια σειρών και προσθέτουμε σε αυτό μια στήλη με το όνομα episode_id, η οποία συνδέεται με τον πίνακα episodes και λειτουργεί ως foreign key. Στον πίνακα αυτό, ελέγχουμε επίσης πως για κάθε εγγραφή του ικανοποιείται μια συνθήκη. Το inventory_id είναι το πρωτεύων κλειδί του πίνακα και με τη λογική πως μία ταινία ή ένα επεισόδιο έχει ένα μοναδικό κωδικό στον κατάλογο, προσθέτουμε τη συνθήκη πως σε μία εγγραφή του πίνακα πρέπει να παραμένει κενό το πεδίο για το film_id ή το episode_id. Για παράδειγμα, όπως φαίνεται και στις παρακάτω εικόνες, όταν προσθέτουμε στον κατάλογο μία σειρά, με inventory_id και film_id, τότε το episode_id **πρέπει** να παραμείνει κενό (NULL). Αντίστοιχα, όταν προσθέτουμε στον κατάλογο ένα επεισόδιο σειράς, με inventory_id και episode_id, τότε το film_id **πρέπει** να παραμείνει άδειο (NULL). Με τη χρήση **CONSTRAINT** καθορίζουμε αυτή τη συνθήκη και αποτρέπουμε

επίσης την προσθήκη εγγραφών όπου ένα film_id και ένα episode_id θα έχουν το ίδιο inventory_id.

TABLE inventory - Εγγραφές για ταινίες

```
MariaDB [tvondemand]> SELECT * FROM inventory LIMIT 5;
+--------------+---------+------------+
| inventory_id | film_id | episode_id |
+--------------+---------+------------+
|            6 |       1 |       NULL |
|           97 |      19 |       NULL |
|          152 |      31 |       NULL |
|          187 |      42 |       NULL |
|          361 |      79 |       NULL |
+--------------+---------+------------+
5 rows in set (0.000 sec)
```

TABLE inventory - Εγγραφές για σειρές

```
|         5151 |    NULL |        755 |
|         5152 |    NULL |        756 |
|         5153 |    NULL |        757 |
|         5154 |    NULL |        758 |
|         5155 |    NULL |        759 |
|         5156 |    NULL |       7510 |
|         5157 |    NULL |       7511 |
|         5158 |    NULL |       7512 |
|         5159 |    NULL |       7513 |
|         5160 |    NULL |       7514 |
|         5161 |    NULL |       7515 |
|         5162 |    NULL |       7516 |
+--------------+---------+------------+
387 rows in set (0.324 sec)
```

Με τη χρήση της εντολής INSERT προσθέτουμε τα επεισόδια που υπάρχουν στον πίνακα episodes, μέσα στον πίνακα inventory κι έτσι, στον πίνακα υπάρχουν τώρα 387 εγγραφές.

Αναφορικά με τους πίνακες rental και payment, δεν κάνουμε οποιεσδήποτε αλλαγές στη δομή τους, αλλά καταχωρούμε νέα αιτήματα ενοικιάσεων και πληρωμές επεισοδίων και ταινιών με ημερομηνίες εντός του 2022.

TABLE rental

```
MariaDB [tvondemand]> SELECT * FROM rental LIMIT 5;
+-----------+---------------------+--------------+-------------+
| rental_id | rental_date         | inventory_id | customer_id |
+-----------+---------------------+--------------+-------------+
|       148 | 2005-05-26 00:25:23 |         4252 |         142 |
|       213 | 2005-05-26 08:44:08 |         1505 |         394 |
|       424 | 2005-05-27 15:34:01 |         2815 |          35 |
|       506 | 2005-05-28 02:09:19 |          667 |         469 |
|       756 | 2005-05-29 10:28:45 |         3152 |         201 |
+-----------+---------------------+--------------+-------------+
5 rows in set (0.000 sec)
```

```
|     15528 | 2005-08-23 03:45:40 |         3033 |         114 |
|     15919 | 2005-08-23 18:01:31 |         1746 |         497 |
|     16040 | 2005-08-23 22:19:33 |         3524 |         195 |
|     12786 | 2006-02-14 15:16:03 |           97 |         512 |
|     12698 | 2006-02-14 15:16:03 |         3657 |         497 |
|     12001 | 2006-02-14 15:16:03 |         4158 |          52 |
|     20001 | 2022-08-22 01:25:08 |         4501 |         602 |
|     20002 | 2022-08-22 01:55:08 |         4502 |         602 |
|     20003 | 2022-08-22 02:35:08 |         4503 |         602 |
|     20004 | 2022-08-22 04:03:13 |         4504 |         602 |
|     20005 | 2022-08-22 07:00:08 |         4505 |         602 |
|     20006 | 2022-08-22 09:00:08 |         4511 |         605 |
|     20008 | 2022-08-22 14:31:09 |         4924 |         626 |
|     20009 | 2022-08-22 16:37:09 |         4917 |         626 |
|    200010 | 2022-08-22 16:48:09 |         4720 |         618 |
+-----------+---------------------+--------------+-------------+
103 rows in set (0.021 sec)
```

TABLE payment

```
MariaDB [tvondemand]> SELECT * FROM payment LIMIT 5;
+------------+-------------+-----------+--------+---------------------+
| payment_id | customer_id | rental_id | amount | payment_date        |
+------------+-------------+-----------+--------+---------------------+
|        434 |          16 |      8452 |   5.99 | 2005-07-29 07:45:00 |
|        910 |          33 |      4095 |   5.99 | 2005-07-07 06:01:48 |
|        922 |          33 |     13958 |   7.99 | 2005-08-20 18:11:44 |
|        924 |          33 |     14623 |   0.99 | 2005-08-21 18:29:13 |
|        953 |          35 |       424 |   6.99 | 2005-05-27 15:34:01 |
+------------+-------------+-----------+--------+---------------------+
5 rows in set (0.000 sec)
```

```
|      14892 |          556 |      1083 |   3.99 | 2005-05-31 11:04:48 |
|      14895 |          556 |      2986 |   0.99 | 2005-06-20 08:50:28 |
|      14909 |          556 |     14176 |   2.99 | 2005-08-21 03:09:23 |
|      15623 |          583 |      5462 |   0.99 | 2005-07-09 22:56:53 |
|      15633 |          583 |     10800 |   4.99 | 2005-08-01 22:07:44 |
|      15967 |          596 |      6197 |   4.99 | 2005-07-11 12:09:51 |
|      15980 |          596 |     14417 |   0.99 | 2005-08-21 11:13:35 |
|      16001 |          602 |     20001 |   0.20 | 2022-08-22 01:25:08 |
|      16002 |          602 |     20002 |   0.20 | 2022-08-22 01:55:08 |
|      16003 |          602 |     20003 |   0.20 | 2022-08-22 02:35:08 |
|      16004 |          602 |     20004 |   0.20 | 2022-08-22 04:03:13 |
|      16005 |          602 |     20005 |   0.20 | 2022-08-22 07:00:08 |
|      16006 |          605 |     20006 |   0.10 | 2022-08-22 09:00:08 |
|      16007 |          626 |     20008 |   0.10 | 2022-08-22 14:31:09 |
|      16008 |          626 |     20009 |   0.10 | 2022-08-22 16:37:09 |
|      16009 |          618 |    200010 |   0.20 | 2022-08-22 16:48:09 |
+------------+--------------+-----------+--------+---------------------+
103 rows in set (0.001 sec)
```

Στο Μέρος Α' της περιγραφής αναφέρονται οι διάφοροι τύποι εγγραφής για τους πελάτες, όπως επίσης και το κόστους για ενοικίαση επεισοδίου σειράς ή ταινίας για κάθε πελάτες ανάλογα με τη συνδρομή που έχει. Στον πίνακα subscription_fees καταχωρούμε το κόστος ενοικίασης έτσι όπως έχει οριστεί από την εταιρία. Στη συνέχεια χρησιμοποιούμε το συγκεκριμένο πίνακα για τις ανάλογες χρεώσεις κατά την ενοικίαση ενός επεισοδίου σειράς ή μιας ταινίας.

TABLE subscription_fees

```
MariaDB [tvondemand]> SELECT * FROM subscription_fees LIMIT 5;
+-------------------+----------+------------+
| subscription_code | film_fee | series_fee |
+-------------------+----------+------------+
|                 1 |     0.40 |       0.00 |
|                 2 |     0.00 |       0.20 |
|                 3 |     0.30 |       0.10 |
+-------------------+----------+------------+
3 rows in set (0.000 sec)
```

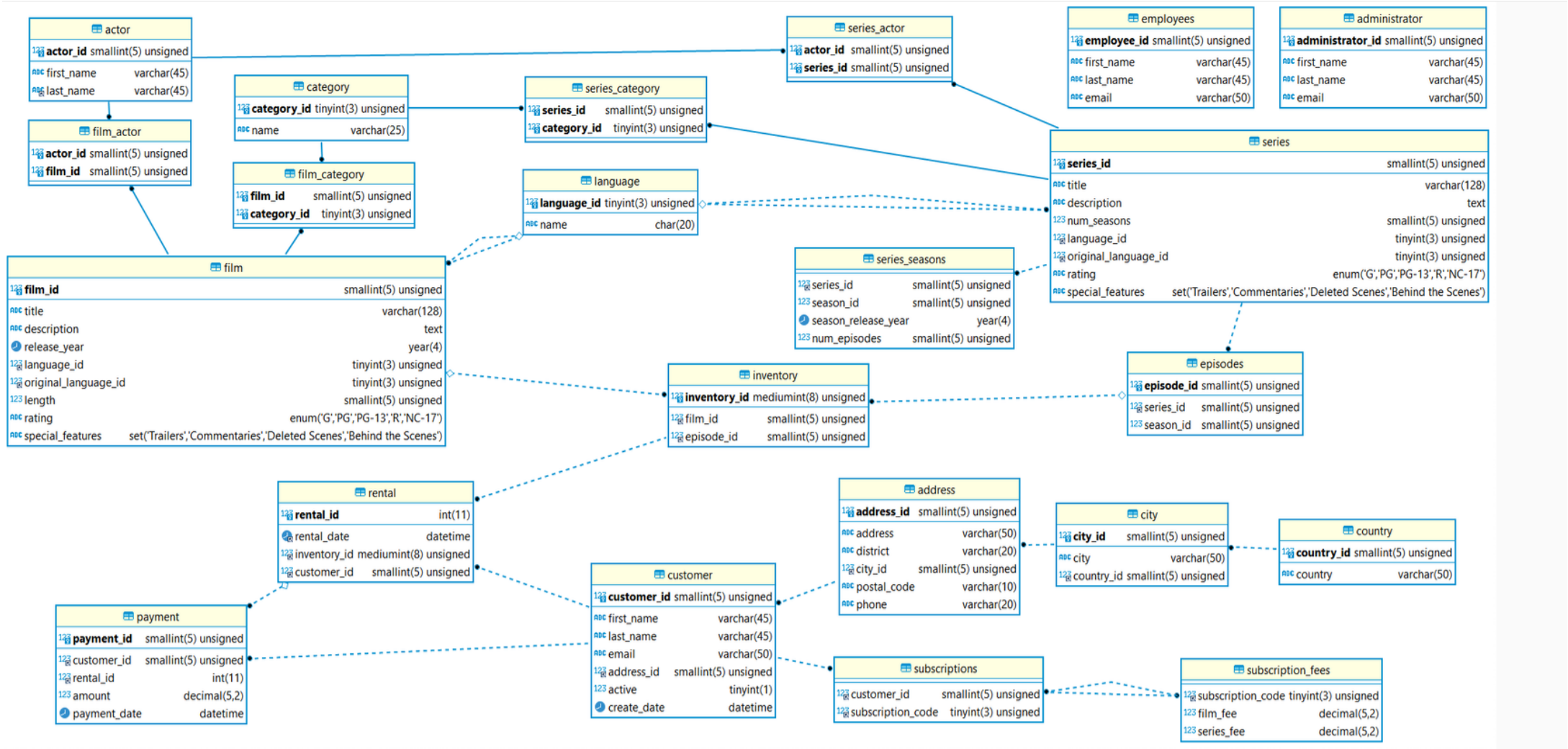
Επιπρόσθετα, δημιουργούμε τον πίνακα subscriptions ο οποίος χρησιμοποιείται για να καταχωρείται σε αυτόν ο τύπος εγγραφής / συνδρομής κάθε πελάτη.

TABLE subscriptions

```
MariaDB [tvondemand]> SELECT * FROM subscriptions LIMIT 5;
+-------------+-------------------+
| customer_id | subscription_code |
+-------------+-------------------+
|           3 |                 1 |
|          16 |                 1 |
|          33 |                 1 |
|          35 |                 1 |
|          52 |                 1 |
+-------------+-------------------+
5 rows in set (0.000 sec)
```

Με τις προσθήκες πινάκων, τις επεκτάσεις και τροποποιήσεις στους υφιστάμενους πίνακες που έχουμε κάνει στη βάση δεδομένων, το σχεσιακό διάγραμμα (ER Diagram) της συνολικής αναθεωρημένης βάσης δεδομένων (ΒΔ tvondemand) διαμορφώνεται ως εξής:

Στο πιο πάνω διάγραμμα, βλέπουμε τις σχέσεις που έχουν διαμορφωθεί μεταξύ των πινάκων που υπάρχουν στη βάση δεδομένων.
Για τη δημιουργία της βάσης δεδομένων χρησιμοποιήσαμε την εφαρμογή XAMPP Control Panel, όπου ενεργοποιήσαμε τον server για να γράψουμε σε MySQL στο περιβάλλον της MariaDB, ενώ για τη γραφική απεικόνιση της ΒΔ χρησιμοποιήσαμε το λογισμικό DBeaver.


actor
actor_id smallint(5) unsigned
first_name varchar(45)
last_name varchar(45)
film_actor
actor_id smallint(5) unsigned
film_id smallint(5) unsigned
category
category_id tinyint(3) unsigned
name varchar(25)
film_category
film_id smallint(5) unsigned
category_id tinyint(3) unsigned
series_actor
actor_id smallint(5) unsigned
series_id smallint(5) unsigned
series_category
series_id smallint(5) unsigned
category_id tinyint(3) unsigned
language
language_id tinyint(3) unsigned
name char(20)
employees
employee_id smallint(5) unsigned
first_name varchar(45)
last_name varchar(45)
email varchar(50)
administrator
administrator_id smallint(5) unsigned
first_name varchar(45)
last_name varchar(45)
email varchar(50)
series
series_id smallint(5) unsigned
title varchar(128)
description text
num_seasons smallint(5) unsigned
language_id tinyint(3) unsigned
original_language_id tinyint(3) unsigned
rating enum('G','PG','PG-13','R','NC-17')
special_features set('Trailers','Commentaries','Deleted Scenes','Behind the Scenes')
film
film_id smallint(5) unsigned
title varchar(128)
description text
release_year year(4)
language_id tinyint(3) unsigned
original_language_id tinyint(3) unsigned
length smallint(5) unsigned
rating enum('G','PG','PG-13','R','NC-17')
special_features set('Trailers','Commentaries','Deleted Scenes','Behind the Scenes')
series_seasons
series_id smallint(5) unsigned
season_id smallint(5) unsigned
season_release_year year(4)
num_episodes smallint(5) unsigned
episodes
episode_id smallint(5) unsigned
series_id smallint(5) unsigned
season_id smallint(5) unsigned
inventory
inventory_id mediumint(8) unsigned
film_id smallint(5) unsigned
episode_id smallint(5) unsigned
rental
rental_id int(11)
rental_date datetime
inventory_id mediumint(8) unsigned
customer_id smallint(5) unsigned
address
address_id smallint(5) unsigned
address varchar(50)
district varchar(20)
city_id smallint(5) unsigned
postal_code varchar(10)
phone varchar(20)
city
city_id smallint(5) unsigned
city varchar(50)
country_id smallint(5) unsigned
country
country_id smallint(5) unsigned
country varchar(50)
payment
payment_id smallint(5) unsigned
customer_id smallint(5) unsigned
rental_id int(11)
amount decimal(5,2)
payment_date datetime
customer
customer_id smallint(5) unsigned
first_name varchar(45)
last_name varchar(45)
email varchar(50)
address_id smallint(5) unsigned
active tinyint(1)
create_date datetime
subscriptions
customer_id smallint(5) unsigned
subscription_code tinyint(3) unsigned
subscription_fees
subscription_code tinyint(3) unsigned
film_fee decimal(5,2)
series_fee decimal(5,2)

Κεφάλαιο 3: Stored Procedures - Κώδικας & Παραδείγματα

3.1 Stored Procedure:

Κώδικας:

```
DELIMITER $

 CREATE PROCEDURE proc_ex_3_1  (
IN film_or_series VARCHAR(1),
IN top_n SMALLINT(5),
IN min_date DATE,
IN max_date DATE
)
BEGIN
   CASE film_or_series
      WHEN 'm' THEN
         SELECT
            F.film_id,
            F.title,
            COUNT(*) as total_rent
         FROM film F
         INNER JOIN inventory I ON F.film_id = I.film_id
         INNER JOIN rental R ON R.inventory_id = I.inventory_id
         INNER JOIN payment P ON P.rental_id = R.rental_id
         WHERE P.payment_date >= min_date AND P.payment_date <=
max_date
         OR P.payment_date <= min_date AND P.payment_date >= max_date
         GROUP BY F.film_id, F.title
         ORDER BY total_rent DESC, film_id ASC
         LIMIT top_n;
        WHEN 's' THEN
         SELECT
            S.series_id,
            S.title,
            COUNT(*) as total_rent
         FROM series S
         INNER JOIN episodes E ON S.series_id = E.series_id
         INNER JOIN inventory I ON I.episode_id = E.episode_id
         INNER JOIN rental R ON R.inventory_id = I.inventory_id
         INNER JOIN payment P ON P.rental_id = R.rental_id
         WHERE P.payment_date >= min_date AND P.payment_date <=
max_date
         OR P.payment_date <= min_date AND P.payment_date >= max_date
         GROUP BY S.series_id, S.title
         ORDER BY total_rent DESC, series_id ASC
         LIMIT top_n;
        END CASE;
END $
```

Παραδείγματα από την εκτέλεση:

CALL proc_ex_3_1('m', 10, '2005-07-31', '2022-07-01');

```
MariaDB [tvondemand]> CALL proc_ex_3_1('m', 10, '2005-07-31', '2022-07-01');
+---------+--------------------+------------+
| film_id | title              | total_rent |
+---------+--------------------+------------+
|     859 | SUGAR WONKA        |          3 |
|     292 | EXCITEMENT EVE     |          2 |
|     379 | GREEDY ROOTS       |          2 |
|     902 | TRADING PINOCCHIO  |          2 |
|      19 | AMADEUS HOLY       |          1 |
|      79 | BLADE POLISH       |          1 |
|     112 | CALENDAR GUNFIGHT  |          1 |
|     172 | CONEHEADS SMOOCHY  |          1 |
|     173 | CONFESSIONS MAGUIRE |          1 |
|     200 | CURTAIN VIDEOTAPE  |          1 |
+---------+--------------------+------------+
10 rows in set (0.240 sec)
```

CALL proc_ex_3_1('m', 6, '2005-07-01', '2022-07-31');

```
MariaDB [tvondemand]> CALL proc_ex_3_1('m', 6, '2005-07-01', '2022-07-31');
+---------+-------------------+------------+
| film_id | title             | total_rent |
+---------+-------------------+------------+
|     859 | SUGAR WONKA       |          3 |
|     902 | TRADING PINOCCHIO |          3 |
|     218 | DECEIVER BETRAYED |          2 |
|     292 | EXCITEMENT EVE    |          2 |
|     379 | GREEDY ROOTS      |          2 |
|     903 | TRAFFIC HOBBIT    |          2 |
+---------+-------------------+------------+
6 rows in set (0.109 sec)
```

CALL proc_ex_3_1('s', 8, '2005-07-01', '2022-09-30');

```
MariaDB [tvondemand]> CALL proc_ex_3_1('s', 8, '2005-07-01', '2022-09-30');
+-----------+-----------------+------------+
| series_id | title           | total_rent |
+-----------+-----------------+------------+
|      1001 | The Sandman     |          6 |
|      1005 | The Boys        |          2 |
|      1003 | Stranger Things |          1 |
+-----------+-----------------+------------+
3 rows in set (0.087 sec)
```

3.2 Stored Procedure:

Κώδικας:

```
DELIMITER $

CREATE PROCEDURE proc_ex_3_2
(
IN given_email VARCHAR(50),
IN given_date DATE,
OUT total_items SMALLINT(5)
)
BEGIN
   SELECT COUNT(*) INTO total_items
      FROM CUSTOMER C
      INNER JOIN SUBSCRIPTIONS S ON C.customer_id = S.customer_id
      INNER JOIN payment P ON C.customer_id = P.customer_id
      WHERE C.email = given_email AND DATE(P.payment_date) = given_date
      GROUP BY C.email, S.subscription_code
      ORDER BY total_items DESC;

   SELECT
      DATE(P.payment_date) as payment_date,
      C.email,
      S.subscription_code,
      COUNT(*) as total_items
   FROM CUSTOMER C
   INNER JOIN SUBSCRIPTIONS S ON C.customer_id = S.customer_id
   INNER JOIN payment P ON C.customer_id = P.customer_id
   WHERE C.email = given_email AND DATE(P.payment_date) = given_date
   GROUP BY C.email, S.subscription_code;

END $
```

Παραδείγματα από την εκτέλεση:

CALL proc_ex_3_2('DAN.PAINE@sakilacustomer.org', '2005-07-07', @total_items);
SELECT @total_items;

```
MariaDB [tvondemand]> CALL proc_ex_3_2('DAN.PAINE@sakilacustomer.org', '2005-07-07', @total_items);
+--------------+------------------------------+-------------------+-------------+
| payment_date | email                        | subscription_code | total_items |
+--------------+------------------------------+-------------------+-------------+
| 2005-07-07   | DAN.PAINE@sakilacustomer.org |                 1 |           2 |
+--------------+------------------------------+-------------------+-------------+
1 row in set (0.002 sec)

Query OK, 1 row affected (0.008 sec)

MariaDB [tvondemand]> SELECT @total_items;
+--------------+
| @total_items |
+--------------+
|            2 |
+--------------+
1 row in set (0.000 sec)
```

CALL proc_ex_3_2('PENELOPE.BARBER@sakilacustomer.org', '2022-08-22', @total_items);
SELECT @total_items;

```
MariaDB [tvondemand]> CALL proc_ex_3_2('PENELOPE.BARBER@sakilacustomer.org', '2022-08-22', @total_items);
+--------------+-----------------------------------+-------------------+-------------+
| payment_date | email                             | subscription_code | total_items |
+--------------+-----------------------------------+-------------------+-------------+
| 2022-08-22   | PENELOPE.BARBER@sakilacustomer.org |                 2 |           5 |
+--------------+-----------------------------------+-------------------+-------------+
1 row in set (0.001 sec)

Query OK, 1 row affected (0.008 sec)

MariaDB [tvondemand]> SELECT @total_items;
+--------------+
| @total_items |
+--------------+
|            5 |
+--------------+
1 row in set (0.000 sec)
```

CALL proc_ex_3_2('GILBERT.SLEDGE@sakilacustomer.org', '2006-02-14', @total_items);
SELECT @total_items;

```
MariaDB [tvondemand]> CALL proc_ex_3_2('GILBERT.SLEDGE@sakilacustomer.org', '2006-02-14', @total_items);
+--------------+-----------------------------------+-------------------+-------------+
| payment_date | email                             | subscription_code | total_items |
+--------------+-----------------------------------+-------------------+-------------+
| 2006-02-14   | GILBERT.SLEDGE@sakilacustomer.org |                 1 |           1 |
+--------------+-----------------------------------+-------------------+-------------+
1 row in set (0.001 sec)

Query OK, 1 row affected (0.008 sec)

MariaDB [tvondemand]> SELECT @total_items;
+--------------+
| @total_items |
+--------------+
|            1 |
+--------------+
1 row in set (0.000 sec)
```

3.3 Stored Procedure

Κώδικας :

```
DELIMITER $
CREATE PROCEDURE proc_ex_3_3()
BEGIN
   SELECT
      YEAR(P.payment_date)  AS payment_date_year,
      DATE_FORMAT(P.payment_date, '%M')  AS payment_date_month,
      SUM(SF.series_fee *
               (CASE WHEN I.episode_id  IS NULL AND        SF.subscription_code
        != 1 THEN 0
               ELSE 1 END)) AS total_series_amount,
      CASE
         WHEN YEAR(P.payment_date) < '2022' THEN SUM(P.amount)
         WHEN YEAR(P.payment_date) >= '2022' THEN SUM(SF.film_fee * (CASE
WHEN                          I.film_id IS NULL THEN 0 ELSE 1 END))
         ELSE NULL
      END as total_film_amount
   FROM payment P
   INNER JOIN customer C ON P.customer_id = C.customer_id
   INNER JOIN subscriptions  S ON C.customer_id = S.customer_id
   INNER JOIN subscription_fees  SF ON S.subscription_code  =
SF.subscription_code
   INNER JOIN rental R ON P.rental_id  = R.rental_id
   INNER JOIN inventory I ON R.inventory_id = I.inventory_id
   GROUP BY payment_date_year,  payment_date_month
   ORDER BY payment_date_year  ASC, payment_date_month  DESC;

END $
```

Παραδείγματα από την εκτέλεση:

CALL proc_ex_3_3();

```
MariaDB [tvondemand]> CALL proc_ex_3_3();
+-------------------+--------------------+---------------------+-------------------+
| payment_date_year | payment_date_month | total_series_amount | total_film_amount |
+-------------------+--------------------+---------------------+-------------------+
|              2005 | May                |                0.00 |             44.89 |
|              2005 | June               |                0.00 |             66.84 |
|              2005 | July               |                0.00 |            170.66 |
|              2005 | August             |                0.00 |            126.70 |
|              2006 | February           |                0.00 |             10.97 |
|              2022 | August             |                1.50 |              0.00 |
+-------------------+--------------------+---------------------+-------------------+
6 rows in set (0.003 sec)
```

3.4 Stored Procedure

Λόγω του μεγάλου πλήθους εγγραφών, δημιουργούμε ένα επιπλέον ευρετήριο με βάση τη στήλη last_name του πίνακα actor. Η προσθήκη ενός ευρετηρίου θα μας βοηθήσει στη βελτιστοποίηση του ερωτήματος (query) αλλά και στην ταχύτητα της συλλογής και παρουσίασης των δεδομένων από τον πίνακα actor.
Δημιουργούμε το ευρετήριο idx_surname και ο τύπος του ευρετηρίου είναι B-Tree με δεδομένο πως χρησιμοποιούμε storage engine InnoDB για όλους τους πίνακες της βάσης δεδομένων μας.

```
--- Add indexing to actor table
CREATE INDEX idx_surname  ON actor(last_name);
SHOW INDEXES FROM actor;
```

```
MariaDB [tvondemand]> SHOW INDEXES FROM actor;
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table | Non_unique | Key_name    | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| actor |          0 | PRIMARY     |            1 | actor_id    | A         |       10293 |     NULL | NULL   |      | BTREE      |         |               |
| actor |          1 | idx_surname |            1 | last_name   | A         |        2573 |     NULL | NULL   |      | BTREE      |         |               |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
2 rows in set (0.002 sec)
```

3.4A Stored Procedure

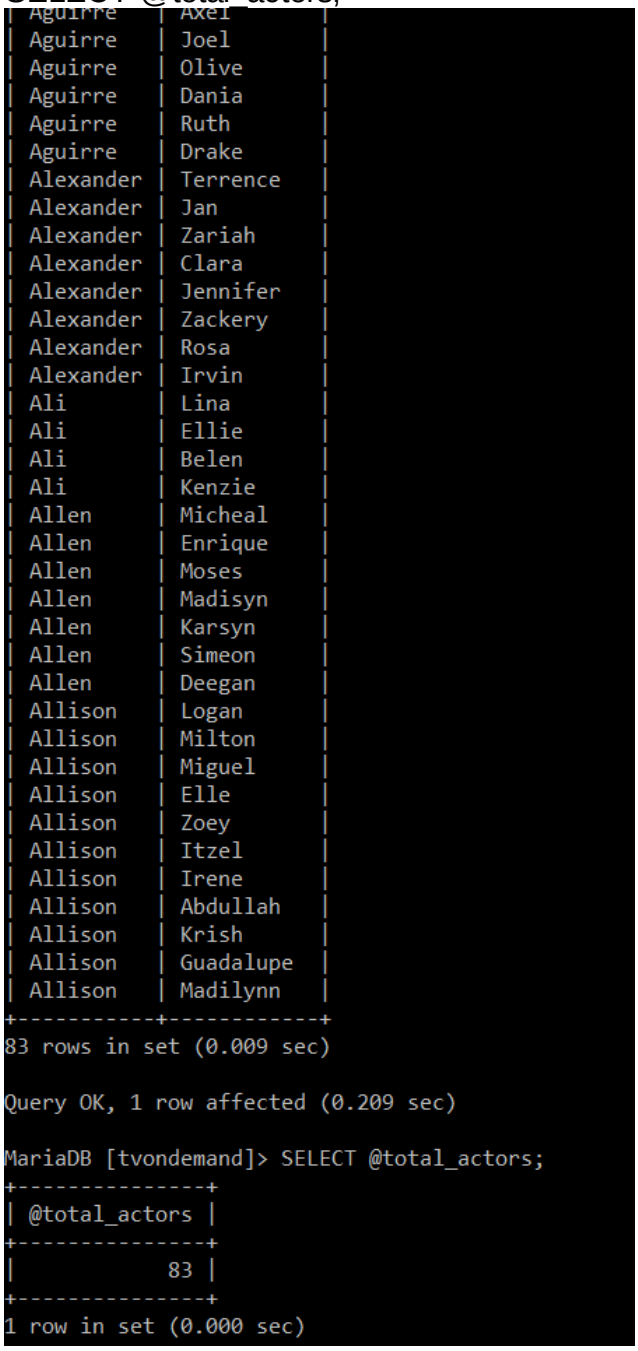
Κώδικας:

```
DELIMITER $

CREATE PROCEDURE proc_ex_3_4a
(
IN min_surname VARCHAR(50),
IN max_surname VARCHAR(50),
OUT total_actors  SMALLINT(5)
)
BEGIN
        SELECT COUNT(*) INTO total_actors
      FROM actor A
      WHERE LEFT(A.last_name,  LENGTH(min_surname)) >= min_surname
                AND LEFT(A.last_name,  LENGTH(min_surname)) <=
                max_surname
      OR LEFT(A.last_name,  LENGTH(min_surname)) <= min_surname
                AND LEFT(A.last_name,  LENGTH(min_surname)) >=
        max_surname;

   SELECT
      A.last_name,
      A.first_name
   FROM actor A
   WHERE LEFT(A.last_name,  LENGTH(min_surname)) >= min_surname
                AND LEFT(A.last_name,  LENGTH(min_surname)) <= max_surname
   OR LEFT(A.last_name,  LENGTH(min_surname)) <= min_surname
        AND LEFT(A.last_name,  LENGTH(min_surname)) >= max_surname
```

```
    ORDER BY A.last_name  ASC;

END $
```

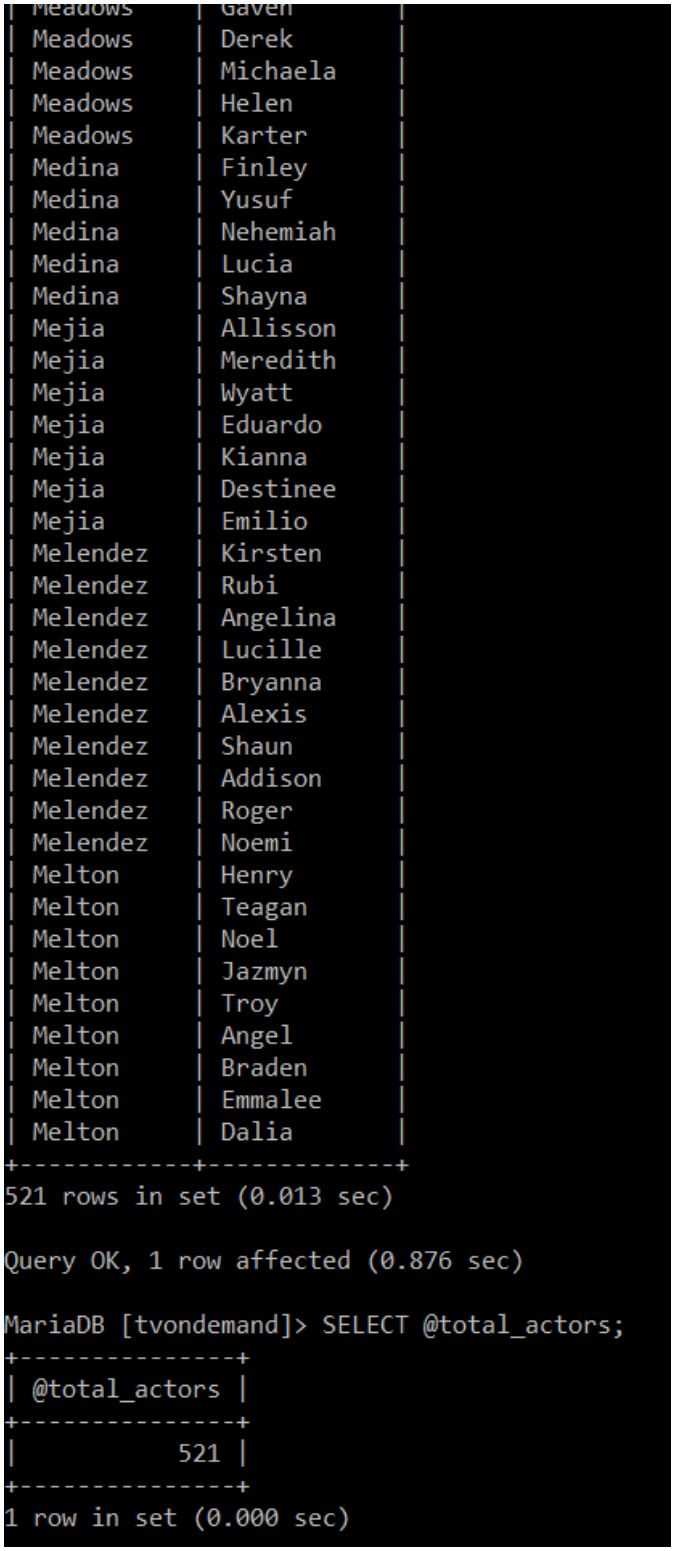
Παραδείγματα από την εκτέλεση :

```
CALL proc_ex_3_4a('Aco', 'Alm', @total_actors);
SELECT @total_actors;
```



```
| Aguirre   | Axel       |
| Aguirre   | Joel       |
| Aguirre   | Olive      |
| Aguirre   | Dania      |
| Aguirre   | Ruth       |
| Aguirre   | Drake      |
| Alexander | Terrence   |
| Alexander | Jan        |
| Alexander | Zariah     |
| Alexander | Clara      |
| Alexander | Jennifer   |
| Alexander | Zackery    |
| Alexander | Rosa       |
| Alexander | Irvin      |
| Ali       | Lina       |
| Ali       | Ellie      |
| Ali       | Belen      |
| Ali       | Kenzie     |
| Allen     | Micheal    |
| Allen     | Enrique    |
| Allen     | Moses      |
| Allen     | Madisyn    |
| Allen     | Karsyn     |
| Allen     | Simeon     |
| Allen     | Deegan     |
| Allison   | Logan      |
| Allison   | Milton     |
| Allison   | Miguel     |
| Allison   | Elle       |
| Allison   | Zoey       |
| Allison   | Itzel      |
| Allison   | Irene      |
| Allison   | Abdullah   |
| Allison   | Krish      |
| Allison   | Guadalupe  |
| Allison   | Madilynn   |
+-----------+------------+
83 rows in set (0.009 sec)

Query OK, 1 row affected (0.209 sec)

MariaDB [tvondemand]> SELECT @total_actors;
+---------------+
| @total_actors |
+---------------+
|            83 |
+---------------+
1 row in set (0.000 sec)
```

CALL proc_ex_3_4a('Mar', 'Mem', @total_actors);
SELECT @total_actors;



```
| Meadows    | Gaven       |
| Meadows    | Derek       |
| Meadows    | Michaela    |
| Meadows    | Helen       |
| Meadows    | Karter      |
| Medina     | Finley      |
| Medina     | Yusuf       |
| Medina     | Nehemiah    |
| Medina     | Lucia       |
| Medina     | Shayna      |
| Mejia      | Allisson    |
| Mejia      | Meredith    |
| Mejia      | Wyatt       |
| Mejia      | Eduardo     |
| Mejia      | Kianna      |
| Mejia      | Destinee    |
| Mejia      | Emilio      |
| Melendez   | Kirsten     |
| Melendez   | Rubi        |
| Melendez   | Angelina    |
| Melendez   | Lucille     |
| Melendez   | Bryanna     |
| Melendez   | Alexis      |
| Melendez   | Shaun       |
| Melendez   | Addison     |
| Melendez   | Roger       |
| Melendez   | Noemi       |
| Melton     | Henry       |
| Melton     | Teagan      |
| Melton     | Noel        |
| Melton     | Jazmyn      |
| Melton     | Troy        |
| Melton     | Angel       |
| Melton     | Braden      |
| Melton     | Emmalee     |
| Melton     | Dalia       |
+------------+-------------+
521 rows in set (0.013 sec)

Query OK, 1 row affected (0.876 sec)

MariaDB [tvondemand]> SELECT @total_actors;
+---------------+
| @total_actors |
+---------------+
|           521 |
+---------------+
1 row in set (0.000 sec)
```

CALL proc_ex_3_4a('BAA', 'BAJ', @total_actors);
SELECT @total_actors;

```
MariaDB [tvondemand]> CALL proc_ex_3_4a('BAA', 'BAJ', @total_actors);
+-----------+------------+
| last_name | first_name |
+-----------+------------+
| Backley   | Hugo       |
| Bailey    | Kaliyah    |
| Bailey    | Jazlynn    |
| Bailey    | Easton     |
| Bailey    | Laylah     |
| Bailey    | Felix      |
| Bailey    | Talan      |
| Baird     | Kiara      |
| Baird     | Keyla      |
| Baird     | Janet      |
| Baird     | Leslie     |
| Baird     | Penelope   |
| Baird     | Erin       |
| Baird     | Slade      |
| Baird     | Cristal    |
| Baird     | Kathleen   |
| Baird     | Gretchen   |
| Baird     | Bryanna    |
| Baird     | Omari      |
+-----------+------------+
19 rows in set (0.011 sec)

Query OK, 1 row affected (0.039 sec)

MariaDB [tvondemand]> SELECT @total_actors;
+---------------+
| @total_actors |
+---------------+
|            19 |
+---------------+
1 row in set (0.000 sec)
```

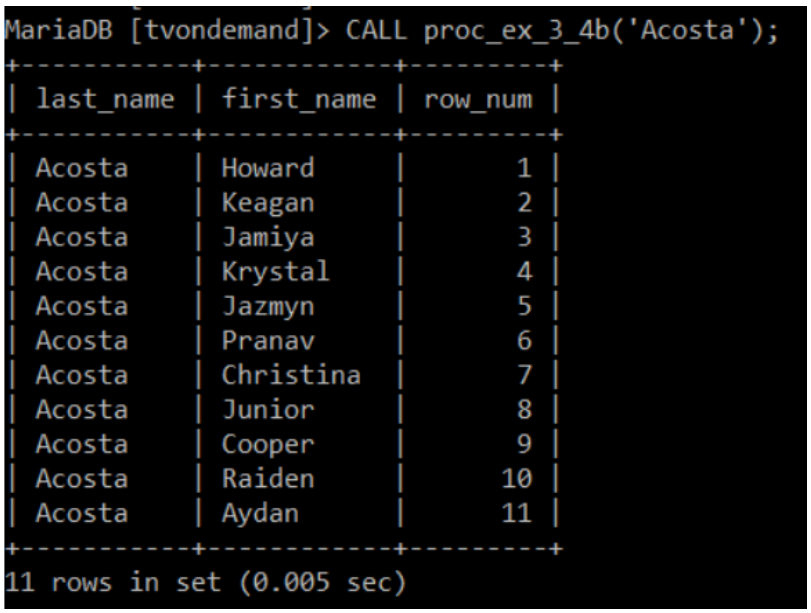
3.4B Stored Procedure

Κώδικας :

```
DELIMITER $
 CREATE PROCEDURE proc_ex_3_4b
(
IN given_surname  VARCHAR(50)
)
BEGIN
   SELECT
      A.last_name,
      A.first_name,
      ROW_NUMBER() OVER (PARTITION BY A.last_name) as row_num
   FROM actor A
   WHERE A.last_name = given_surname
   ORDER BY A.last_name;

END $
```

Παραδείγματα από την εκτέλεση:

```
CALL proc_ex_3_4b('Acosta');
```



```
MariaDB [tvondemand]> CALL proc_ex_3_4b('Acosta');
+-----------+------------+---------+
| last_name | first_name | row_num |
+-----------+------------+---------+
| Acosta    | Howard     |       1 |
| Acosta    | Keagan     |       2 |
| Acosta    | Jamiya     |       3 |
| Acosta    | Krystal    |       4 |
| Acosta    | Jazmyn     |       5 |
| Acosta    | Pranav     |       6 |
| Acosta    | Christina  |       7 |
| Acosta    | Junior     |       8 |
| Acosta    | Cooper     |       9 |
| Acosta    | Raiden     |      10 |
| Acosta    | Aydan      |      11 |
+-----------+------------+---------+
11 rows in set (0.005 sec)
```

CALL proc_ex_3_4b('Hayden');

```
MariaDB [tvondemand]> CALL proc_ex_3_4b('Hayden')
+-----------+------------+---------+
| last_name | first_name | row_num |
+-----------+------------+---------+
| Hayden    | Kade       |       1 |
| Hayden    | Maximus    |       2 |
| Hayden    | Zavier     |       3 |
| Hayden    | Devin      |       4 |
| Hayden    | Maggie     |       5 |
| Hayden    | Franklin   |       6 |
| Hayden    | Larissa    |       7 |
| Hayden    | Emily      |       8 |
| Hayden    | Adelyn     |       9 |
| Hayden    | Fisher     |      10 |
| Hayden    | Morgan     |      11 |
| Hayden    | Guillermo  |      12 |
| Hayden    | Sara       |      13 |
| Hayden    | Dillan     |      14 |
| Hayden    | Liliana    |      15 |
| Hayden    | Stephen    |      16 |
| Hayden    | Zackary    |      17 |
| Hayden    | Rose       |      18 |
| Hayden    | Moises     |      19 |
| Hayden    | Gerald     |      20 |
| Hayden    | Evan       |      21 |
| Hayden    | Carina     |      22 |
+-----------+------------+---------+
22 rows in set (0.004 sec)
```

CALL proc_ex_3_4b('Murray');

```
MariaDB [tvondemand]> CALL proc_ex_3_4b('Murray');
+-----------+------------+---------+
| last_name | first_name | row_num |
+-----------+------------+---------+
| Murray    | Jenny      |       1 |
| Murray    | Travis     |       2 |
| Murray    | Skyla      |       3 |
| Murray    | Laura      |       4 |
| Murray    | Roderick   |       5 |
| Murray    | Kameron    |       6 |
| Murray    | Dakota     |       7 |
| Murray    | Korbin     |       8 |
| Murray    | Journey    |       9 |
| Murray    | Maritza    |      10 |
| Murray    | Craig      |      11 |
| Murray    | Cory       |      12 |
| Murray    | Elise      |      13 |
| Murray    | Mauricio   |      14 |
| Murray    | Beatrice   |      15 |
| Murray    | Angie      |      16 |
| Murray    | Valery     |      17 |
| Murray    | Kenny      |      18 |
| Murray    | Krista     |      19 |
| Murray    | Vaughn     |      20 |
| Murray    | Mareli     |      21 |
+-----------+------------+---------+
21 rows in set (0.006 sec)
```

Κώδικας SQL – CREATE

```
DROP SCHEMA IF EXISTS tvondemand;
CREATE SCHEMA tvondemand;
USE tvondemand;
--
-- Table structure for table `actor`
--
CREATE TABLE actor (
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  PRIMARY KEY  (actor_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `country`
--
CREATE TABLE country (
  country_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  country VARCHAR(50) NOT NULL,
  PRIMARY KEY  (country_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `city`
--
CREATE TABLE city (
  city_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  city VARCHAR(50) NOT NULL,
  country_id SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY  (city_id),
  CONSTRAINT `fk_city_country` FOREIGN KEY (country_id) REFERENCES country (country_id)
ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `address`
--
CREATE TABLE address (
  address_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  address VARCHAR(50) NOT NULL,
  district VARCHAR(20) DEFAULT NULL,
  city_id SMALLINT UNSIGNED NOT NULL,
  postal_code VARCHAR(10) DEFAULT NULL,
  phone VARCHAR(20) NOT NULL,
  PRIMARY KEY  (address_id),
  CONSTRAINT `fk_address_city` FOREIGN KEY (city_id) REFERENCES city (city_id) ON
DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `category`
```

```
--
CREATE TABLE category (
  category_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
  name VARCHAR(25) NOT NULL,
  PRIMARY KEY  (category_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `language`
--
CREATE TABLE language (
  language_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
  name CHAR(20) NOT NULL,
  PRIMARY KEY (language_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `customer`
--
CREATE TABLE customer (
  customer_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  email VARCHAR(50) DEFAULT NULL,
  address_id SMALLINT UNSIGNED NOT NULL,
  active BOOLEAN NOT NULL DEFAULT TRUE,
  create_date DATETIME NOT NULL,
  PRIMARY KEY  (customer_id),
  CONSTRAINT fk_customer_address FOREIGN KEY (address_id) REFERENCES address
(address_id) ON DELETE RESTRICT ON UPDATE CASCADE
  )ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `film`
--
CREATE TABLE film (
  film_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  title VARCHAR(128) NOT NULL,
  description TEXT DEFAULT NULL,
  release_year YEAR DEFAULT NULL,
  language_id TINYINT UNSIGNED NOT NULL,
  original_language_id TINYINT UNSIGNED DEFAULT NULL,
  length SMALLINT UNSIGNED DEFAULT NULL,
  rating ENUM('G','PG','PG-13','R','NC-17') DEFAULT 'G',
  special_features SET('Trailers','Commentaries','Deleted Scenes','Behind the Scenes') DEFAULT
NULL,
  PRIMARY KEY  (film_id),
  CONSTRAINT fk_film_language FOREIGN KEY (language_id) REFERENCES language
(language_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_film_language_original FOREIGN KEY (original_language_id) REFERENCES
language (language_id) ON DELETE RESTRICT ON UPDATE CASCADE
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `series`
--
CREATE TABLE series (
  series_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  title VARCHAR(128) NOT NULL,
  description TEXT DEFAULT NULL,
  num_seasons SMALLINT UNSIGNED NOT NULL,
  language_id TINYINT UNSIGNED NOT NULL,
  original_language_id TINYINT UNSIGNED DEFAULT NULL,
  rating ENUM('G','PG','PG-13','R','NC-17') DEFAULT 'G',
  special_features SET('Trailers','Commentaries','Deleted Scenes','Behind the Scenes') DEFAULT
NULL,
  PRIMARY KEY  (series_id),
  CONSTRAINT fk_series_language FOREIGN KEY (language_id) REFERENCES language
(language_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_series_language_original FOREIGN KEY (original_language_id)
REFERENCES language (language_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `series_seasons`
--
CREATE TABLE series_seasons (
  series_id SMALLINT UNSIGNED NOT NULL,
  season_id SMALLINT UNSIGNED NOT NULL,
  season_release_year YEAR DEFAULT NULL,
  num_episodes SMALLINT UNSIGNED NOT NULL,
  CONSTRAINT fk_series_seasons_series_id FOREIGN KEY (series_id) REFERENCES series
(series_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `episodes`
--
CREATE TABLE episodes (
  episode_id SMALLINT UNSIGNED NOT NULL,
  series_id SMALLINT UNSIGNED NOT NULL,
  season_id SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY (episode_id),
  CONSTRAINT fk_episodes_series_id FOREIGN KEY (series_id) REFERENCES series
(series_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `film_actor`
--
CREATE TABLE film_actor (
  actor_id SMALLINT UNSIGNED NOT NULL,
  film_id SMALLINT UNSIGNED NOT NULL,
```

```
  PRIMARY KEY  (actor_id,film_id),
  CONSTRAINT fk_film_actor_actor FOREIGN KEY (actor_id) REFERENCES actor (actor_id) ON
DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_film_actor_film FOREIGN KEY (film_id) REFERENCES film (film_id) ON
DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `series_actor`
--
CREATE TABLE series_actor (
  actor_id SMALLINT UNSIGNED NOT NULL,
  series_id SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY  (actor_id, series_id),
  CONSTRAINT fk_series_actor_actor FOREIGN KEY (actor_id) REFERENCES actor (actor_id)
ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_series_actor_series FOREIGN KEY (series_id) REFERENCES series
(series_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `film_category`
--
CREATE TABLE film_category (
  film_id SMALLINT UNSIGNED NOT NULL,
  category_id TINYINT UNSIGNED NOT NULL,
  PRIMARY KEY (film_id, category_id),
  CONSTRAINT fk_film_category_film FOREIGN KEY (film_id) REFERENCES film (film_id) ON
DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_film_category_category FOREIGN KEY (category_id) REFERENCES category
(category_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
 --
-- Table structure for table `series_category`
--
 CREATE TABLE series_category (
  series_id SMALLINT UNSIGNED NOT NULL,
  category_id TINYINT UNSIGNED NOT NULL,
  PRIMARY KEY (series_id, category_id),
  CONSTRAINT fk_series_category_series FOREIGN KEY (series_id) REFERENCES series
(series_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_series_category_category FOREIGN KEY (category_id) REFERENCES
category (category_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table `inventory`
--
CREATE TABLE inventory (
  inventory_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
  film_id SMALLINT UNSIGNED,
  episode_id SMALLINT UNSIGNED,
  PRIMARY KEY  (inventory_id),
  CONSTRAINT fk_inventory_film FOREIGN KEY (film_id) REFERENCES film (film_id) ON
DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_inventory_episode FOREIGN KEY (episode_id) REFERENCES episodes
(episode_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT chk CHECK (
    film_id IS NOT NULL AND episode_id IS NULL OR
    film_id IS NULL AND episode_id IS NOT NULL)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `rental`
--
CREATE TABLE rental (
  rental_id INT NOT NULL AUTO_INCREMENT,
  rental_date DATETIME NOT NULL,
  inventory_id MEDIUMINT UNSIGNED NOT NULL,
  customer_id SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY (rental_id),
  UNIQUE KEY  (rental_date,inventory_id,customer_id),
  CONSTRAINT fk_rental_inventory FOREIGN KEY (inventory_id) REFERENCES inventory
(inventory_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_rental_customer FOREIGN KEY (customer_id) REFERENCES customer
(customer_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table `payment`
--
CREATE TABLE payment (
  payment_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  customer_id SMALLINT UNSIGNED NOT NULL,
  rental_id INT DEFAULT NULL,
  amount DECIMAL(5,2) NOT NULL,
  payment_date DATETIME NOT NULL,
  PRIMARY KEY  (payment_id),
  CONSTRAINT fk_payment_rental FOREIGN KEY (rental_id) REFERENCES rental (rental_id) ON
DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT fk_payment_customer FOREIGN KEY (customer_id) REFERENCES customer
(customer_id) ON DELETE RESTRICT ON UPDATE CASCADE
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `subscriptions`
--
 CREATE TABLE subscriptions (
  customer_id SMALLINT UNSIGNED NOT NULL,
  subscription_code TINYINT UNSIGNED NOT NULL,
```

```
  CONSTRAINT fk_subscripticons_customer FOREIGN KEY (customer_id) REFERENCES
customer (customer_id) ON DELETE RESTRICT ON UPDATE CASCADE
  -- , CONSTRAINT fk_subscriptions_subscription_code FOREIGN KEY (subscription_code)
REFERENCES subscription_fees (subscription_code) ON DELETE RESTRICT ON UPDATE
CASCADE
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `subscription_fees`
--
CREATE TABLE subscription_fees (
  subscription_code TINYINT UNSIGNED NOT NULL,
  film_fee DECIMAL(5,2) NOT NULL,
  series_fee DECIMAL(5,2) NOT NULL
  -- , CONSTRAINT fk_subscription_fees_subscription FOREIGN KEY (subscription_code)
REFERENCES subscriptions (subscription_code) ON DELETE RESTRICT ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `employees`
--
CREATE TABLE employees (
  employee_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  email VARCHAR(50) DEFAULT NULL,
  PRIMARY KEY (employee_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `administrator`
--
CREATE TABLE administrator (
  administrator_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  email VARCHAR(50) DEFAULT NULL,
  PRIMARY KEY (administrator_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- Table structure for table `log_table`
--
CREATE TABLE log_table (
  username VARCHAR(255),
  user_type VARCHAR(45),
  action_type VARCHAR(45),
  action_table VARCHAR(45),
  action_datetime_ DATETIME,
  if_successful BOOLEAN
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Κώδικας SQL – INSERT

Προσθέτουμε ΜΟΝΟ τις δικές μας εντολές INSERT λόγω του μεγάλου όγκου των εντολών

```
--
-- Dumping data for table subscriptions
--
INSERT INTO `subscriptions` (`customer_id`, `subscription_code`) VALUES
(3, 1),
(16, 1),
(33, 1),
(35, 1),
(52, 1),
(85, 1),
(104, 1),
(114, 1),
(142, 1),
(162, 1),
(195, 1),
(196, 1),
(201, 1),
(221, 1),
(227, 1),
(251, 1),
(252, 1),
(293, 1),
(309, 1),
(394, 1),
(402, 1),
(439, 1),
(448, 1),
(469, 1),
(477, 1),
```

(497, 1),

(499, 1),

(512, 1),

(525, 1),

(541, 1),

(549, 1),

(556, 1),

(583, 1),

(596, 1);

--

-- Dumping data for table customer

--

INSERT INTO `customer` (`customer_id`, `first_name`, `last_name`, `email`, `address_id`, `active`, `create_date`) VALUES

(602, 'PENELOPE', 'BARBER', 'PENELOPE.BARBER@sakilacustomer.org', 407, 1, '2022-08-20 12:00:00'),

(603, 'KAYLEY', 'SULLIVAN', 'KAYLEY.SULLIVAN@sakilacustomer.org', 231, 1, '2022-08-20 12:00:00'),

(605, 'EMANUEL', 'LOWERY', 'EMANUEL.LOWERY@sakilacustomer.org', 314, 1, '2022-08-20 12:00:00'),

(606, 'JANE', 'TRUJILLO', 'JANE.TRUJILLO@sakilacustomer.org', 56, 1, '2022-08-20 12:00:00'),

(608, 'BRITTANY', 'MCDOWELL', 'BRITTANY.MCDOWELL@sakilacustomer.org', 399, 1, '2022-08-20 12:00:00'),

(611, 'JOSELYN', 'MOLINA', 'JOSELYN.MOLINA@sakilacustomer.org', 166, 1, '2022-08-20 12:00:00'),

(613, 'JOEY', 'BARTON', 'JOEY.BARTON@sakilacustomer.org', 255, 1, '2022-08-20 12:00:00'),

(615, 'WESLEY', 'SAMPSON', 'WESLEY.SAMPSON@sakilacustomer.org', 199, 1, '2022-08-20 12:00:00'),

(618, 'BREANNA', 'HODGES', 'BREANNA.HODGES@sakilacustomer.org', 555, 1, '2022-08-20 12:00:00'),

(619, 'SKYE', 'ALVARADO', 'SKYE.ALVARADO@sakilacustomer.org', 20, 1, '2022-08-20 12:00:00'),

(622, 'MADISYN', 'HATFIELD', 'MADISYN.HATFIELD@sakilacustomer.org', 562, 1, '2022-08-20 12:00:00'),

(626, 'MAXIMILIAN', 'VASQUEZ', 'MAXIMILIAN.VASQUEZ@sakilacustomer.org', 482, 1, '2022-08-20 12:00:00'),

(628, 'RODRIGO', 'GALLAGHER', 'RODRIGO.GALLAGHER@sakilacustomer.org', 108, 1, '2022-08-20 12:00:00'),

(632, 'GAVYN', 'TAYLOR', 'GAVYN.TAYLOR@sakilacustomer.org', 531, 1, '2022-08-20 12:00:00'),

(635, 'DEMARCUS', 'CISNEROS', 'DEMARCUS.CISNEROS@sakilacustomer.org', 205, 1, '2022-08-20 12:00:00'),

(638, 'KAILEY', 'SERRANO', 'KAILEY.SERRANO@sakilacustomer.org', 502, 1, '2022-08-20 12:00:00'),

(643, 'COLIN', 'WOODARD', 'COLIN.WOODARD@sakilacustomer.org', 37, 1, '2022-08-20 12:00:00'),

(645, 'AMELIE', 'TANNER', 'AMELIE.TANNER@sakilacustomer.org', 89, 1, '2022-08-20 12:00:00'),

(659, 'WINCENT', 'GONZALES', 'WINCENT.GONZALES@sakilacustomer.org', 444, 1, '2022-08-20 12:00:00'),

(668, 'SHAYLEE', 'SHORT', 'SHAYLEE.SHORT@sakilacustomer.org', 118, 1, '2022-08-20 12:00:00');

--

-- Dumping data for table subscriptions

--

INSERT INTO `subscriptions` (`customer_id`, `subscription_code`) VALUES

(602, 2),

(603, 2),

(605, 3),

(606, 3),

(608, 1),

(611, 2),

(613, 2),

(615, 2),

(618, 2),

(619, 2),

(622, 3),

(626, 3),

(628, 3),

(632, 2),

```
(635, 2),
(638, 2),
(643, 3),
(645, 3),
(659, 3),
(668, 2);
--
-- Dumping data for table subscription_fees
--
INSERT INTO `subscription_fees` (`subscription_code`, `film_fee`, `series_fee`) VALUES
(1, 0.4, 0),
(2, 0, 0.2),
(3, 0.3, 0.1);
--
-- Dumping data for table employees
--
INSERT INTO `employees` (`employee_id`, `first_name`, `last_name`, `email`) VALUES
(123, "Alexander", "Green", "alex.green@sakilaemployee.org"),
(124, "Athena", "Pappa", "athena.pappa@sakilaemployee.org"),
(125, "Andreas", "Eleftherakis", 'andreas.eleftherakis@sakilaemployee.org'),
(126, "George", "Ioannou", 'george.ioannou@sakilaemployee.org'),
(127, "Periklis", "Smith", "periklis.smith@sakilaemployee.org");
--
-- Dumping data for table administrator
--
INSERT INTO `administrator` (`administrator_id`, `first_name`, `last_name`, `email`) VALUES
(211, "Elena", "Vlachou", "elena.vlachou@sakiladmin.org"),
(212, "Nikos", "Pappas", "nikos.pappas@sakiladmin.org"),
(213, "Andreas", "Iacovou", "andreas.iacovou@sakiladmin.org");
```

--

-- Dumping data for table series

--

INSERT INTO `series` (`series_id`, `title`, `description`, `num_seasons`, `language_id`, `original_language_id`, `rating`, `special_features`) VALUES

(1001, 'The Sandman', 'Upon escaping after decades of imprisonment by a mortal wizard, Dream, the personification of dreams, sets about to reclaim his lost equipment', 1, 1, NULL, 'G', 'Deleted Scenes'),

(1002, 'Better Call Saul', 'The trials and tribulations of criminal lawyer Jimmy McGill before his fateful run-in with Walter White and Jesse Pinkman', 6, 1, 1, NULL, NULL),

(1003, 'Stranger Things', 'When a young boy disappears, his mother, a police chief and his friends must confront terrifying supernatural forces in order to get him back', 4, 1, 1, 'G', 'Behind the Scenes'),

(1004, 'Game of Thrones', 'Nine noble families fight for control over the lands of Westeros, while an ancient enemy returns after being dormant for millennia', 8, 1, NULL, 'G', NULL),

(1005, 'The Boys', 'A group of vigilantes set out to take down corrupt superheroes who abuse their superpowers', 3, 1, 1, 'G', 'Commentaries'),

(1006, 'Westworld', 'At the intersection of the near future and the reimagined past, waits a world in which every human appetite can be indulged without consequence', 4, 1, NULL, 'PG-13', NULL),

(1007, 'Breaking Bad', 'A high school chemistry teacher diagnosed with inoperable lung cancer turns to manufacturing and selling methamphetamine in order to secure his familys future', 5, 1, 1, 'G', NULL);

--

-- Dumping data for table series_seasons

--

INSERT INTO `series_seasons` (`series_id`, `season_id`, `season_release_year`, `num_episodes`) VALUES

(1001, 1, 2022, 11),

(1002, 1, 2015, 10),

(1002, 2, 2016, 10),

(1002, 3, 2017, 10),

(1002, 4, 2018, 10),

(1002, 5, 2020, 10),

(1002, 6, 2022, 13),

(1003, 1, 2016, 8),

(1003, 2, 2017, 9),

(1003, 3, 2019, 8),

(1003, 4, 2022, 9),

(1004, 1, 2011, 10),

(1004, 2, 2012, 10),

(1004, 3, 2013, 10),

(1004, 4, 2014, 10),

(1004, 5, 2015, 10),

(1004, 6, 2016, 10),

(1004, 7, 2017, 7),

(1004, 8, 2019, 6),

(1005, 1, 2019, 8),

(1005, 2, 2020, 8),

(1005, 3, 2022, 8),

(1006, 1, 2016, 10),

(1006, 2, 2018, 10),

(1006, 3, 2020, 8),

(1006, 4, 2022, 8),

(1007, 1, 2008, 7),

(1007, 2, 2009, 13),

(1007, 3, 2010, 13),

(1007, 4, 2011, 13),

(1007, 5, 2012, 16);

--

-- Dumping data for table episodes

--

INSERT INTO `episodes` (`episode_id`, `series_id`, `season_id`) VALUES

(111, 1001, 1), (112, 1001, 1), (113, 1001, 1), (114, 1001, 1), (115, 1001, 1),

(116, 1001, 1), (117, 1001, 1), (118, 1001, 1), (119, 1001, 1), (1110, 1001, 1), (1111, 1001, 1),

(211, 1002, 1), (212, 1002, 1), (213, 1002, 1), (214, 1002, 1), (215, 1002, 1),

(216, 1002, 1), (217, 1002, 1), (218, 1002, 1), (219, 1002, 1), (2110, 1002, 1),

(221, 1002, 2), (222, 1002, 2), (223, 1002, 2), (224, 1002, 2), (225, 1002, 2),

(226, 1002, 2), (227, 1002, 2), (228, 1002, 2), (229, 1002, 2), (2210, 1002, 2),
(231, 1002, 3), (232, 1002, 3), (233, 1002, 3), (234, 1002, 3), (235, 1002, 3),
(236, 1002, 3), (237, 1002, 3), (238, 1002, 3), (239, 1002, 3), (2310, 1002, 3),
(241, 1002, 4), (242, 1002, 4), (243, 1002, 4), (244, 1002, 4), (245, 1002, 4),
(246, 1002, 4), (247, 1002, 4), (248, 1002, 4), (249, 1002, 4), (2410, 1002, 4),
(251, 1002, 5), (252, 1002, 5), (253, 1002, 5), (254, 1002, 5), (255, 1002, 5),
(256, 1002, 5), (257, 1002, 5), (258, 1002, 5), (259, 1002, 5), (2510, 1002, 5),
(261, 1002, 6), (262, 1002, 6), (263, 1002, 6), (264, 1002, 6), (265, 1002, 6),
(266, 1002, 6), (267, 1002, 6), (268, 1002, 6), (269, 1002, 6), (2610, 1002, 6),
(2611, 1002, 6), (2612, 1002, 6), (2613, 1002, 6),
(311, 1003, 1), (312, 1003, 1), (313, 1003, 1), (314, 1003, 1),
(315, 1003, 1), (316, 1003, 1), (317, 1003, 1), (318, 1003, 1),
(321, 1003, 2), (322, 1003, 2), (323, 1003, 2), (324, 1003, 2),
(325, 1003, 2), (326, 1003, 2), (327, 1003, 2), (328, 1003, 2), (329, 1003, 2),
(331, 1003, 3), (332, 1003, 3), (333, 1003, 3), (334, 1003, 3),
(335, 1003, 3), (336, 1003, 3), (337, 1003, 3), (338, 1003, 3),
(341, 1003, 4), (342, 1003, 4), (343, 1003, 4), (344, 1003, 4),
(345, 1003, 4), (346, 1003, 4), (347, 1003, 4), (348, 1003, 4), (349, 1003, 4),
(411, 1004, 1), (412, 1004, 1), (413, 1004, 1), (414, 1004, 1), (415, 1004, 1),
(416, 1004, 1), (417, 1004, 1), (418, 1004, 1), (419, 1004, 1), (4110, 1004, 1),
(421, 1004, 2), (422, 1004, 2), (423, 1004, 2), (424, 1004, 2), (425, 1004, 2),
(426, 1004, 2), (427, 1004, 2), (428, 1004, 2), (429, 1004, 2), (4210, 1004, 2),
(431, 1004, 3), (432, 1004, 3), (433, 1004, 3), (434, 1004, 3), (435, 1004, 3),
(436, 1004, 3), (437, 1004, 3), (438, 1004, 3), (439, 1004, 3), (4310, 1004, 3),
(441, 1004, 4), (442, 1004, 4), (443, 1004, 4), (444, 1004, 4), (445, 1004, 4),
(446, 1004, 4), (447, 1004, 4), (448, 1004, 4), (449, 1004, 4), (4410, 1004, 4),
(451, 1004, 5), (452, 1004, 5), (453, 1004, 5), (454, 1004, 5), (455, 1004, 5),
(456, 1004, 5), (457, 1004, 5), (458, 1004, 5), (459, 1004, 5), (4510, 1004, 5),
(461, 1004, 6), (462, 1004, 6), (463, 1004, 6), (464, 1004, 6), (465, 1004, 6),
(466, 1004, 6), (467, 1004, 6), (468, 1004, 6), (469, 1004, 6), (4610, 1004, 6),
(471, 1004, 7), (472, 1004, 7), (473, 1004, 7), (474, 1004, 7), (475, 1004, 7),

(476, 1004, 7), (477, 1004, 7),

(481, 1004, 8), (482, 1004, 8), (483, 1004, 8),

(484, 1004, 8), (485, 1004, 8), (486, 1004, 8),

(511, 1005, 1), (512, 1005, 1), (513, 1005, 1), (514, 1005, 1),

(515, 1005, 1), (516, 1005, 1), (517, 1005, 1), (518, 1005, 1),

(521, 1005, 2), (522, 1005, 2), (523, 1005, 2), (524, 1005, 2),

(525, 1005, 2), (526, 1005, 2), (527, 1005, 2), (528, 1005, 2),

(531, 1005, 3), (532, 1005, 3), (533, 1005, 3), (534, 1005, 3),

(535, 1005, 3), (536, 1005, 3), (537, 1005, 3), (538, 1005, 3),

(611, 1006, 1), (612, 1006, 1), (613, 1006, 1), (614, 1006, 1), (615, 1006, 1),

(616, 1006, 1), (617, 1006, 1), (618, 1006, 1), (619, 1006, 1), (6110, 1006, 1),

(621, 1006, 2), (622, 1006, 2), (623, 1006, 2), (624, 1006, 2), (625, 1006, 2),

(626, 1006, 2), (627, 1006, 2), (628, 1006, 2), (629, 1006, 2), (6210, 1006, 2),

(631, 1006, 3), (632, 1006, 3), (633, 1006, 3), (634, 1006, 3),

(635, 1006, 3), (636, 1006, 3), (637, 1006, 3), (638, 1006, 3),

(641, 1006, 4), (642, 1006, 4), (643, 1006, 4), (644, 1006, 4),

(645, 1006, 4), (646, 1006, 4), (647, 1006, 4), (648, 1006, 4),

(711, 1007, 1), (712, 1007, 1), (713, 1007, 1), (714, 1007, 1),

(715, 1007, 1), (716, 1007, 1), (717, 1007, 1),

(721, 1007, 2), (722, 1007, 2), (723, 1007, 2), (724, 1007, 2), (725, 1007, 2),

(726, 1007, 2), (727, 1007, 2), (728, 1007, 2), (729, 1007, 2), (7210, 1007, 2),

(7211, 1007, 2), (7212, 1007, 2), (7213, 1007, 2),

(731, 1007, 3), (732, 1007, 3), (733, 1007, 3), (734, 1007, 3), (735, 1007, 3),

(736, 1007, 3), (737, 1007, 3), (738, 1007, 3), (739, 1007, 3), (7310, 1007, 3),

(7311, 1007, 3), (7312, 1007, 3), (7313, 1007, 3),

(741, 1007, 4), (742, 1007, 4), (743, 1007, 4), (744, 1007, 4), (745, 1007, 4),

(746, 1007, 4), (747, 1007, 4), (748, 1007, 4), (749, 1007, 4), (7410, 1007, 4),

(7411, 1007, 4), (7412, 1007, 4), (7413, 1007, 4),

(751, 1007, 5), (752, 1007, 5), (753, 1007, 5), (754, 1007, 5),

(755, 1007, 5), (756, 1007, 5), (757, 1007, 5), (758, 1007, 5),

(759, 1007, 5), (7510, 1007, 5), (7511, 1007, 5), (7512, 1007, 5),

```
(7513, 1007, 5), (7514, 1007, 5), (7515, 1007, 5), (7516, 1007, 5);
--
-- Dumping data for table actor
--
INSERT INTO `actor` (`actor_id`, `first_name`, `last_name`) VALUES
(10011,'TOM','STURRIDGE'),
(10012,'BOYD','HOLBROOK'),
(10013,'PATTON','OSWALT'),
(10014, 'BOB', 'ODENKIRK'),
(10015, 'RHEA', 'SEEHORN'),
(10016, 'JONATHAN', 'BANKS'),
(10017, 'MILLIE', 'BOBBY BROWN'),
(10018, 'FINN', 'WOLFHARD'),
(10019, 'WINONA', 'RYDER'),
(10020, 'EMILIA', 'CLARKE'),
(10021, 'PETER', 'DINKLAGE'),
(10022, 'KIT', 'HARINGTON'),
(10023, 'KARL', 'URBAN'),
(10024, 'JACK', 'QUAID'),
(10025, 'ANTONY', 'STARR'),
(10026, 'EVAN', 'RACHEL'),
(10027, 'WOOD', 'JEFFREY WRIGHT'),
(10028, 'ED', 'HARIS'),
(10029, 'BRYAN', 'CRANSTON'),
(10030, 'AARON', 'PAUL'),
(10031, 'ANNA', 'GUNN');
--
-- Dumping data for table series_actor
--
INSERT INTO `series_actor` (`actor_id`, `series_id`) VALUES
(10011, 1001),
```

(10012, 1001),

(10013, 1001),

(10014, 1002),

(10015, 1002),

(10016, 1002),

(10017, 1003),

(10018, 1003),

(10019, 1003),

(10020, 1004),

(10021, 1004),

(10022, 1004),

(10023, 1005),

(10024, 1005),

(10025, 1005),

(10026, 1006),

(10027, 1006),

(10028, 1006),

(10029, 1007),

(10030, 1007),

(10031, 1007);

--

-- Dumping data for table series_category

--

INSERT INTO `series_category` (`series_id`, `category_id`) VALUES

(1001, 11),

(1002, 7),

(1003, 7),

(1004, 1),

(1005, 1),

(1006, 14),

(1007, 7);

--

-- Dumping data for table inventory

--

INSERT INTO `inventory` (`inventory_id`, `film_id`, `episode_id`) VALUES

(4501, NULL, 111), (4502, NULL, 112), (4503, NULL, 113), (4504, NULL, 114), (4505, NULL, 115),

(4506, NULL, 116), (4507, NULL, 117), (4508, NULL, 118), (4509, NULL, 119), (4510, NULL, 1110), (4511, NULL, 1111),

(4601, NULL, 211), (4602, NULL, 212), (4603, NULL, 213), (4604, NULL, 214), (4605, NULL, 215),

(4606, NULL, 216), (4607, NULL, 217), (4608, NULL, 218), (4609, NULL, 219), (4610, NULL, 2110),

(4611, NULL, 221), (4612, NULL, 222), (4613, NULL, 223), (4614, NULL, 224), (4615, NULL, 225),

(4616, NULL, 226), (4617, NULL, 227), (4618, NULL, 228), (4619, NULL, 229), (4620, NULL, 2210),

(4621, NULL, 231), (4622, NULL, 232), (4623, NULL, 233), (4624, NULL, 234), (4625, NULL, 235),

(4626, NULL, 236), (4627, NULL, 237), (4628, NULL, 238), (4629, NULL, 239), (4630, NULL, 2310),

(4631, NULL, 241), (4632, NULL, 242), (4633, NULL, 243), (4634, NULL, 244), (4635, NULL, 245),

(4636, NULL, 246), (4637, NULL, 247), (4638, NULL, 248), (4639, NULL, 249), (4640, NULL, 2410),

(4641, NULL, 251), (4642, NULL, 252), (4643, NULL, 253), (4644, NULL, 254), (4645, NULL, 255),

(4646, NULL, 256), (4647, NULL, 257), (4648, NULL, 258), (4649, NULL, 259), (4650, NULL, 2510),

(4651, NULL, 261), (4652, NULL, 262), (4653, NULL, 263), (4654, NULL, 264), (4655, NULL, 265),

(4656, NULL, 266), (4657, NULL, 267), (4658, NULL, 268), (4659, NULL, 269), (4660, NULL, 2610),

(4661, NULL, 2611), (4662, NULL, 2612), (4663, NULL, 2613),

(4701, NULL, 311), (4702, NULL, 312), (4703, NULL, 313), (4704, NULL, 314),

(4705, NULL, 315), (4706, NULL, 316), (4707, NULL, 317), (4708, NULL, 318),

(4709, NULL, 321), (4710, NULL, 322), (4711, NULL, 323), (4712, NULL, 324),

(4713, NULL, 325), (4714, NULL, 326), (4715, NULL, 327), (4716, NULL, 328), (4717, NULL, 329),

(4718, NULL, 331), (4719, NULL, 332), (4720, NULL, 333), (4721, NULL, 334),

(4722, NULL, 335), (4723, NULL, 336), (4724, NULL, 337), (4725, NULL, 338),

(4726, NULL, 341), (4727, NULL, 342), (4728, NULL, 343), (4729, NULL, 344),

(4730, NULL, 345), (4731, NULL, 346), (4732, NULL, 347), (4733, NULL, 348), (4734, NULL, 349),

(4801, NULL, 411), (4802, NULL, 412), (4803, NULL, 413), (4804, NULL, 414), (4805, NULL, 415),

(4806, NULL, 416), (4807, NULL, 417), (4808, NULL, 418), (4809, NULL, 419), (4810, NULL, 4110),

(4811, NULL, 421), (4812, NULL, 422), (4813, NULL, 423), (4814, NULL, 424), (4815, NULL, 425),

(4816, NULL, 426), (4817, NULL, 427), (4818, NULL, 428), (4819, NULL, 429), (4820, NULL, 4210),

(4821, NULL, 431), (4822, NULL, 432), (4823, NULL, 433), (4824, NULL, 434), (4825, NULL, 435),

(4826, NULL, 436), (4827, NULL, 437), (4828, NULL, 438), (4829, NULL, 439), (4830, NULL, 4310),

(4831, NULL, 441), (4832, NULL, 442), (4833, NULL, 443), (4834, NULL, 444), (4835, NULL, 445),

(4836, NULL, 446), (4837, NULL, 447), (4838, NULL, 448), (4839, NULL, 449), (4840, NULL, 4410),

(4841, NULL, 451), (4842, NULL, 452), (4843, NULL, 453), (4844, NULL, 454), (4845, NULL, 455),

(4846, NULL, 456), (4847, NULL, 457), (4848, NULL, 458), (4849, NULL, 459), (4850, NULL, 4510),

(4851, NULL, 461), (4852, NULL, 462), (4853, NULL, 463), (4854, NULL, 464), (4855, NULL, 465),

(4856, NULL, 466), (4857, NULL, 467), (4858, NULL, 468), (4859, NULL, 469), (4860, NULL, 4610),

(4861, NULL, 471), (4862, NULL, 472), (4863, NULL, 473), (4864, NULL, 474),

(4865, NULL, 475), (4866, NULL, 476), (4867, NULL, 477),

(4868, NULL, 481), (4869, NULL, 482), (4870, NULL, 483),

(4871, NULL, 484), (4872, NULL, 485), (4873, NULL, 486),

(4901, NULL, 511), (4902, NULL, 512), (4903, NULL, 513), (4904, NULL, 514),

(4905, NULL, 515), (4906, NULL, 516), (4907, NULL, 517), (4908, NULL, 518),

(4909, NULL, 521), (4910, NULL, 522), (4911, NULL, 523), (4912, NULL, 524),

(4913, NULL, 525), (4914, NULL, 526), (4915, NULL, 527), (4916, NULL, 528),

(4917, NULL, 531), (4918, NULL, 532), (4919, NULL, 533), (4920, NULL, 534),

(4921, NULL, 535), (4922, NULL, 536), (4923, NULL, 537), (4924, NULL, 538),

(5001, NULL, 611), (5002, NULL, 612), (5003, NULL, 613), (5004, NULL, 614), (5005, NULL, 615),

(5006, NULL, 616), (5007, NULL, 617), (5008, NULL, 618), (5009, NULL, 619), (5010, NULL, 6110),

(5011, NULL, 621), (5012, NULL, 622), (5013, NULL, 623), (5014, NULL, 624), (5015, NULL, 625),

(5016, NULL, 626), (5017, NULL, 627), (5018, NULL, 628), (5019, NULL, 629), (5020, NULL, 6210),

(5021, NULL, 631), (5022, NULL, 632), (5023, NULL, 633), (5024, NULL, 634),

(5025, NULL, 635), (5026, NULL, 636), (5027, NULL, 637), (5028, NULL, 638),

(5029, NULL, 641), (5030, NULL, 642), (5031, NULL, 643), (5032, NULL, 644),

(5033, NULL, 645), (5034, NULL, 646), (5035, NULL, 647),(5036, NULL, 648),

(5101, NULL, 711), (5102, NULL, 712), (5103, NULL, 713), (5104, NULL, 714),

(5105, NULL, 715), (5106, NULL, 716), (5107, NULL, 717),

(5108, NULL, 721), (5109, NULL, 722), (5110, NULL, 723), (5111, NULL, 724), (5112, NULL, 725),

(5113, NULL, 726), (5114, NULL, 727), (5115, NULL, 728), (5116, NULL, 729), (5117, NULL, 7210),

(5118, NULL, 7211), (5119, NULL, 7212), (5120, NULL, 7213),

(5121, NULL, 731), (5122, NULL, 732), (5123, NULL, 733), (5124, NULL, 734), (5125, NULL, 735),

(5126, NULL, 736), (5127, NULL, 737), (5128, NULL, 738), (5129, NULL, 739), (5130, NULL, 7310),

(5131, NULL, 7311), (5132, NULL, 7312), (5133, NULL, 7313),

(5134, NULL, 741), (5135, NULL, 742), (5136, NULL, 743), (5137, NULL, 744), (5138, NULL, 745),

(5139, NULL, 746), (5140, NULL, 747), (5141, NULL, 748), (5142, NULL, 749), (5143, NULL, 7410),

(5144, NULL, 7411), (5145, NULL, 7412), (5146, NULL, 7413),

(5147, NULL, 751), (5148, NULL, 752), (5149, NULL, 753), (5150, NULL, 754),

(5151, NULL, 755), (5152, NULL, 756), (5153, NULL, 757), (5154, NULL, 758),

(5155, NULL, 759), (5156, NULL, 7510), (5157, NULL, 7511), (5158, NULL, 7512),

(5159, NULL, 7513), (5160, NULL, 7514), (5161, NULL, 7515), (5162, NULL, 7516);

--

-- Dumping data for table rental

--

INSERT INTO `rental` (`rental_id`, `rental_date`, `inventory_id`, `customer_id`) VALUES

(20001, '2022-08-22 01:25:08', 4501, 602),

(20002, '2022-08-22 01:55:08', 4502, 602),

(20003, '2022-08-22 02:35:08', 4503, 602),

(20004, '2022-08-22 04:03:13', 4504, 602),

(20005, '2022-08-22 07:00:08', 4505, 602),

(20006, '2022-08-22 09:00:08', 4511, 605);

(20007, '2022-08-22 11:01:13', 1061, 602);

INSERT INTO `rental` (`rental_id`, `rental_date`, `inventory_id`, `customer_id`) VALUES

(20008, '2022-08-22 14:31:09', 4924, 626),

(20009, '2022-08-22 16:37:09', 4917, 626),

(200010, '2022-08-22 16:48:09', 4720, 618);

--

-- Dumping data for table payment

--

INSERT INTO `payment` (`payment_id`, `customer_id`, `rental_id`, `amount`, `payment_date`) VALUES

(16001, 602, 20001, '0.20', '2022-08-22 01:25:08'),

(16002, 602, 20002, '0.20', '2022-08-22 01:55:08'),

(16003, 602, 20003, '0.20', '2022-08-22 02:35:08'),

(16004, 602, 20004, '0.20', '2022-08-22 04:03:13'),

(16005, 602, 20005, '0.20', '2022-08-22 07:00:08'),

(16006, 605, 20006, '0.10', '2022-08-22 09:00:08');

INSERT INTO `payment` (`payment_id`, `customer_id`, `rental_id`, `amount`, `payment_date`) VALUES

(16007, 626, 20008, '0.10', '2022-08-22 14:31:09'),

(16008, 626, 20009, '0.10', '2022-08-22 16:37:09'),

(16009, 618, 200010, '0.20', '2022-08-22 16:48:09');

--

-- Dumping data for table log_table

--

INSERT INTO `log_table` (`username`, `user_type`, `action_type`, `action_table`, `action_datetime_`, `if_successful`) VALUES

('elena.vlachou@sakiladmin.org', 'Administrator', 'update', 'RENTAL', '2022-08-29 12:00:00', 1);

Κώδικας SQL – STORED PROCEDURES

```
--
-- STORED PROCEDURE 3.1
--
DELIMITER $
CREATE PROCEDURE proc_ex_3_1
(
IN film_or_series VARCHAR(1),
IN top_n SMALLINT(5),
IN min_date DATE,
IN max_date DATE
)
BEGIN
   CASE film_or_series
      WHEN 'm' THEN
         SELECT
            F.film_id,
            F.title,
            COUNT(*) as total_rent
         FROM film F
         INNER JOIN inventory I
         ON F.film_id = I.film_id
         INNER JOIN rental R
         ON R.inventory_id = I.inventory_id
         INNER JOIN payment P
         ON P.rental_id = R.rental_id
         WHERE P.payment_date >= min_date AND P.payment_date <= max_date
         OR P.payment_date <= min_date AND P.payment_date >= max_date
         GROUP BY F.film_id, F.title
         ORDER BY total_rent DESC, film_id ASC
         LIMIT top_n;

      WHEN 's' THEN

         SELECT
            S.series_id,
            S.title,
            COUNT(*) as total_rent
         FROM series S
         INNER JOIN episodes E
         ON S.series_id = E.series_id
         INNER JOIN inventory I
         ON I.episode_id = E.episode_id
         INNER JOIN rental R
         ON R.inventory_id = I.inventory_id
         INNER JOIN payment P
         ON P.rental_id = R.rental_id
         WHERE P.payment_date >= min_date AND P.payment_date <= max_date
```

```
        OR P.payment_date <= min_date AND P.payment_date >= max_date
        GROUP BY S.series_id, S.title
        ORDER BY total_rent DESC, series_id ASC
        LIMIT top_n;
    END CASE;

END $

SHOW CREATE PROCEDURE proc_ex_3_1;

CALL proc_ex_3_1('m', 10, '2005-07-31', '2022-07-01');
CALL proc_ex_3_1('m', 6, '2005-07-01', '2022-07-31');
CALL proc_ex_3_1('s', 8, '2005-07-01', '2022-09-30');




--
-- STORED PROCEDURE 3.2
--

DELIMITER $

CREATE PROCEDURE proc_ex_3_2
(
IN given_email VARCHAR(50),
IN given_date DATE,
OUT total_items SMALLINT(5)
)

BEGIN

    SELECT COUNT(*) INTO total_items
        FROM CUSTOMER C
        INNER JOIN SUBSCRIPTIONS S ON C.customer_id = S.customer_id
        INNER JOIN payment P ON C.customer_id = P.customer_id
        WHERE C.email = given_email AND DATE(P.payment_date) = given_date
        GROUP BY C.email, S.subscription_code
        ORDER BY total_items DESC;


    SELECT
        DATE(P.payment_date) as payment_date,
        C.email,
        S.subscription_code,
        COUNT(*) as total_items
    FROM CUSTOMER C
    INNER JOIN SUBSCRIPTIONS S
```

```
    ON C.customer_id = S.customer_id
    INNER JOIN payment P
    ON C.customer_id = P.customer_id
    WHERE C.email = given_email AND DATE(P.payment_date) = given_date
    GROUP BY C.email, S.subscription_code;

END $

SHOW create procedure proc_ex_3_2a;

CALL proc_ex_3_2('DAN.PAINE@sakilacustomer.org', '2005-07-07', @total_items);
SELECT @total_items;
CALL proc_ex_3_2('PENELOPE.BARBER@sakilacustomer.org', '2022-08-22', @total_items);
SELECT @total_items;
CALL proc_ex_3_2('EMANUEL.LOWERY@sakilacustomer.org', '2022-08-22', @total_items);
SELECT @total_items;
CALL proc_ex_3_2('GRACE.ELLIS@sakilacustomer.org', '2005-08-23', @total_items);
SELECT @total_items;
CALL proc_ex_3_2('GILBERT.SLEDGE@sakilacustomer.org', '2006-02-14', @total_items);
SELECT @total_items;
CALL proc_ex_3_2('JULIE.SANCHEZ@sakilacustomer.org', '2006-02-14', @total_items);
SELECT @total_items;



--
-- STORED PROCEDURE 3.3
--


DELIMITER $

CREATE PROCEDURE proc_ex_3_3()

BEGIN

    SELECT
        YEAR(P.payment_date) AS payment_date_year,
        DATE_FORMAT(P.payment_date, '%M')  AS payment_date_month,
        SUM(SF.series_fee * (CASE WHEN I.episode_id IS NULL AND SF.subscription_code != 1
THEN 0 ELSE 1 END)) AS total_series_amount,

        CASE
            WHEN YEAR(P.payment_date) < '2022' THEN SUM(P.amount)
            WHEN YEAR(P.payment_date) >= '2022' THEN SUM(SF.film_fee * (CASE WHEN I.film_id
IS NULL THEN 0 ELSE 1 END))
            ELSE NULL
        END as total_film_amount
```

```
    FROM payment P
    INNER JOIN customer C
    ON P.customer_id = C.customer_id
    INNER JOIN subscriptions S
    ON C.customer_id = S.customer_id
    INNER JOIN subscription_fees SF
    ON S.subscription_code = SF.subscription_code
    INNER JOIN rental R
    ON P.rental_id = R.rental_id
    INNER JOIN inventory I
    ON R.inventory_id = I.inventory_id
    GROUP BY payment_date_year, payment_date_month
    ORDER BY payment_date_year ASC, payment_date_month DESC;

END $

SHOW CREATE PROCEDURE proc_ex_3_3;

CALL proc_ex_3_3();



--
-- STORED PROCEDURE 3.4 - INDEXING OF last_name COLUMN to optimise query and the
speed of data retrieval
--

CREATE INDEX idx_surname  ON actor(last_name);
SHOW INDEXES FROM actor;
--
--



--
-- STORED PROCEDURE 3.4.A
--

DELIMITER $

CREATE PROCEDURE proc_ex_3_4a
(
IN min_surname VARCHAR(50),
IN max_surname VARCHAR(50),
OUT total_actors SMALLINT(5)
)
BEGIN

    SELECT COUNT(*) INTO total_actors
```

```
        FROM actor A
        WHERE LEFT(A.last_name, LENGTH(min_surname)) >= min_surname AND
LEFT(A.last_name, LENGTH(min_surname)) <= max_surname
        OR LEFT(A.last_name, LENGTH(min_surname)) <= min_surname AND LEFT(A.last_name,
LENGTH(min_surname)) >= max_surname;

    SELECT
        A.last_name,
        A.first_name
    FROM actor A
    WHERE LEFT(A.last_name, LENGTH(min_surname)) >= min_surname AND
LEFT(A.last_name, LENGTH(min_surname)) <= max_surname
    OR LEFT(A.last_name, LENGTH(min_surname)) <= min_surname AND LEFT(A.last_name,
LENGTH(min_surname)) >= max_surname
    ORDER BY A.last_name ASC;

END $

SHOW create procedure proc_ex_3_4a;

CALL proc_ex_3_4a('Aco', 'Alm', @total_actors);
SELECT @total_actors;
CALL proc_ex_3_4a('Mar', 'Mem', @total_actors);
SELECT @total_actors;
CALL proc_ex_3_4a('BAA', 'BAJ', @total_actors);
SELECT @total_actors;



--
-- STORED PROCEDURE 3.4.B
--

DELIMITER $

CREATE PROCEDURE proc_ex_3_4b
(
IN given_surname VARCHAR(50)
)
BEGIN

    SELECT
        A.last_name,
        A.first_name,
        ROW_NUMBER() OVER (PARTITION BY A.last_name) as row_num
    FROM actor A
    WHERE A.last_name = given_surname
    ORDER BY A.last_name;
```

```
END $

SHOW create procedure proc_ex_3_4b;

CALL proc_ex_3_4b('Acosta');
CALL proc_ex_3_4b('Hayden');
CALL proc_ex_3_4b('Murray');
```