**Homework Assignment #1**

**Data Management for Data Science**

**Student**

**Vigèr Durand AZIMEDEM TSAFACK (1792126)**

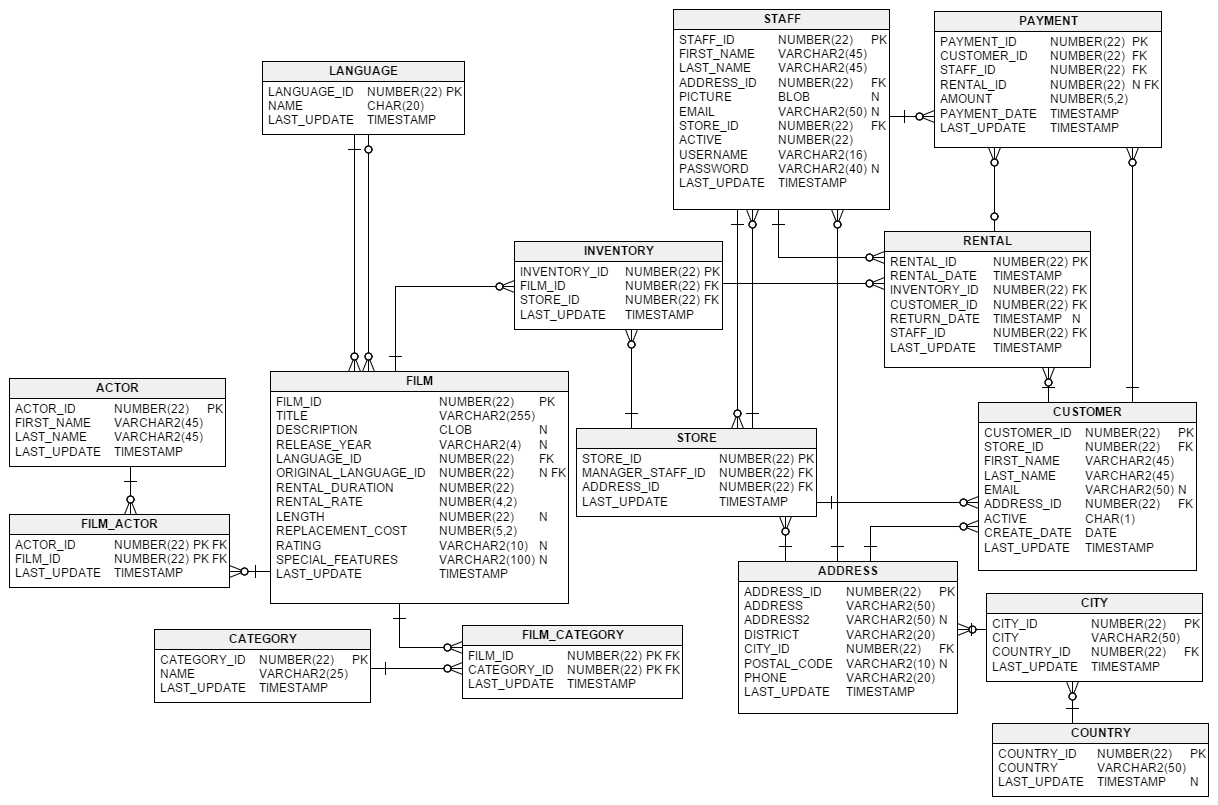
**Data**

We’ll use the **Sakila** schema, which can be found on the following link. <http://dev.mysql.com/doc/index-other.html> ("sakila database")

The MySQL's Sample Salika (DVD Rental) Database is a complex database with 16 tables. It also illustrates features such as Views, Stored Procedures and Triggers.

We will dig into a deep understanding of this database and then we will formulate (and solve) a set of 10 SQL queries on it. We are dealing with a nicely normalized schema modelling a DVD rental store, featuring things like films, actors, film-actor relationships, and a central inventory table that connects films, stores, and rentals.

Follows a graph showing relationships between all the tables of the database:



**Queries**

1. What are the names of all the languages in the database (sorted alphabetically)?
2. **select** l.**name**
3. **from** language l
4. **order** **by** l.**name** **asc**;

|  |
| --- |
| name |
| English |
| Italian |
| Japanese |
| Mandarin |
| French |
| German |

1. Return the full names (first and last) of actors with “SON” in their last name, ordered by their first name.
2. **select** CONCAT(first\_name, ' ', last\_name) **as** full\_name
3. **from** actor
4. -- where last\_name REGEXP '.\*SON.\*'
5. **where** last\_name like '%SON%'
6. **order** **by** first\_name **asc**;

|  |
| --- |
| full\_name |
| ALBERT JOHANSSON |
| ANGELA HUDSON |
| BETTE NICHOLSON |
| CHRISTIAN NEESON |
| JAYNE NEESON |
| MATTHEW JOHANSSON |
| MERYL GIBSON |
| RAY JOHANSSON |
| WILL WILSON |

1. Find all the addresses where the district is not empty (i.e., contains some text), and return these districts sorted.
2. **select** address, district
3. **from** address
4. **where** district **is** not null and district not like ''
5. **order** **by** district;

|  |  |
| --- | --- |
| address | district |
| 669 Firozabad Loop | Abu Dhabi |
| 535 Ahmadnagar Manor | Abu Dhabi |
| 1078 Stara Zagora Drive | Aceh |
| 842 Salzburg Lane | Adana |
| 663 Baha Blanca Parkway | Adana |
| 614 Pak Kret Street | Addis Abeba |
| 751 Lima Loop | Aden |
| 1157 Nyeri Loop | Adygea |
| 387 Mwene-Ditu Drive | Ahal |
| 775 ostka Drive | al-Daqahliya |
| 1759 Niznekamsk Avenue | al-Manama |
| 1152 Citrus Heights Manor | al-Qadarif |
| 1987 Coacalco de Berriozbal Loop | al-Qalyubiya |
| 765 Southampton Drive | al-Qalyubiya |
| 289 Santo Andr Manor | al-Sharqiya |

1. Return the first and last names of actors who played in a film involving a “Crocodile” and a “Shark”, along with the release year of the movie, sorted by the actors’ last names.
2. **select** a.first\_name, a.last\_name, f.release\_year, f.title, f.description
3. **from** film\_actor fa
4. **inner** join film f **on** fa.film\_id = f.film\_id
5. **inner** join actor a **on** fa.actor\_id = a.actor\_id
6. **where** CONCAT(f.title, ' ', f.description) like '%crocodile%'
7. and CONCAT(f.title, ' ', f.description) like '%shark%'
8. **order** **by** a.last\_name **asc**;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| first\_name | last\_name | release\_year | title | description |
| KIRSTEN | AKROYD | 2006 | MADIGAN DORADO | A Astounding Character Study of a A Shark And a A Shark who must Discover a Crocodile in The Outback |
| KIM | ALLEN | 2006 | CLEOPATRA DEVIL | A Fanciful Documentary of a Crocodile And a Technical Writer who must Fight a A Shark in A Baloon |
| AUDREY | BAILEY | 2006 | WARLOCK WEREWOLF | A Astounding Yarn of a Pioneer And a Crocodile who must Defeat a A Shark in The Outback |
| JULIA | BARRYMORE | 2006 | SHOOTIST SUPERFLY | A Fast-Paced Story of a Crocodile And a A Shark who must Sink a Pioneer in Berlin |
| VIVIEN | BASINGER | 2006 | CONNECTICUT TRAMP | A Unbelieveable Drama of a Crocodile And a Mad Cow who must Reach a Dentist in A Shark Tank |
| VIVIEN | BERGEN | 2006 | EXCITEMENT EVE | A Brilliant Documentary of a Monkey And a Car who must Conquer a Crocodile in A Shark Tank |

1. How many films involve a “Crocodile” and a “Shark”?
2. **select** count(\*)
3. **from** film f
4. **where** CONCAT(f.title, ' ', f.description) like '%crocodile%'
5. and CONCAT(f.title, ' ', f.description) like '%shark%';

|  |
| --- |
| count(\*) |
| 10 |

1. What is the average running time of films by category?
2. **select** fc.category\_id, avg(length) **as** avg\_running\_time
3. **from** film f, film\_category fc
4. **where**  f.film\_id = fc.film\_id
5. **group** **by** fc.category\_id;

|  |  |
| --- | --- |
| category\_id | avg\_running\_time |
| 1 | 111.6094 |
| 2 | 111.0152 |
| 3 | 109.8000 |
| 4 | 111.6667 |
| 5 | 115.8276 |
| 6 | 108.7500 |
| 7 | 120.8387 |
| 8 | 114.7826 |
| 9 | 121.6986 |
| 10 | 127.8361 |
| 11 | 112.4821 |
| 12 | 113.6471 |
| 13 | 111.1270 |
| 14 | 108.1967 |
| 15 | 128.2027 |
| 16 | 113.3158 |

1. Which actor has appeared in the most films?
2. **select** a.actor\_id, a.first\_name, a.last\_name, film\_count
3. **from**
4. (**select** actor\_id, count(\*) **as** film\_count
5. **from** film\_actor fa
6. **group** **by** fa.actor\_id) **as** fa **inner** join actor a **on** fa.actor\_id = a.actor\_id
7. **order** **by** film\_count **desc** limit 1;

|  |  |  |  |
| --- | --- | --- | --- |
| actor\_id | first\_name | last\_name | film\_count |
| 107 | GINA | DEGENERES | 42 |

1. When is each copy(inventory) of ‘Academy Dinosaur’ due?
2. **select** inventory\_id, **max**(return\_date)
3. **from** rental
4. **where** inventory\_id in (
5. **select** inventory\_id
6. **from** inventory
7. **where** film\_id = (
8. **select** film\_id
9. **from** film
10. **where** title = 'Academy Dinosaur'
11. )
12. )
13. **group** **by** inventory\_id;

|  |  |
| --- | --- |
| inventory\_id | max(return\_date) |
| 1 | 2005-08-30 22:26:43 |
| 2 | 2005-08-30 20:08:01 |
| 3 | 2005-08-25 18:58:37 |
| 4 | 2005-08-23 21:09:42 |
| 6 | 2005-08-01 04:08:11 |
| 7 | 2005-08-27 02:18:08 |
| 8 | 2005-08-22 22:01:16 |

1. Find all the film categories in which there are between 55 and 65 films. Return the names of these categories and the number of films per category, sorted by the number of films.
2. **select** category.**name** **as** category\_name, film\_count
3. **from** (**select** category\_id, count(\*) **as** film\_count
4. **from** film\_category
5. **group** **by** category\_id) **as** film\_category
6. **inner** join category **on** category.category\_id = film\_category.category\_id
7. **where** film\_count > 55 and film\_count < 65
8. **order** **by** film\_count **asc**;

|  |  |
| --- | --- |
| category\_name | film\_count |
| Horror | 56 |
| Travel | 57 |
| Classics | 57 |
| Comedy | 58 |
| Children | 60 |
| Games | 61 |
| Sci-Fi | 61 |
| Drama | 62 |
| New | 63 |
| Action | 64 |

1. In how many film categories is the average difference between the film replacement cost and the rental rate larger than 17?
2. **select** count(\*)
3. **from** (
4. **select** **name**, avg(replacement\_rental\_diff) **as** avg\_replacement\_rental\_diff
5. **from** (
6. **select** category.**name**, (replacement\_cost - rental\_rate) **as** replacement\_rental\_diff
7. **from** film\_category
8. **inner** join film **on** film.film\_id = film\_category.film\_id
9. **inner** join category **on** category.category\_id = film\_category.category\_id
10. ) **as** replacement
11. **group** **by** **name**
12. ) **as** replacement\_avg
13. **where** avg\_replacement\_rental\_diff > 17

|  |
| --- |
| count(\*) |
| 8 |

**References**

<https://www.ntu.edu.sg/home/ehchua/programming/sql/SampleDatabases.html>

<https://www.jooq.org/sakila>