Great resource for reverse engineering <https://dev.mysql.com/doc/workbench/en/wb-reverse-engineer-live.html>

Also can open in a web browser “/sakila\_schema.svg” file to see the schema

#This is my HomeWork for SQL with all my queries below created:

HW10 - README.md

============================================================================

1a. Display the first and last names of all actors from the table actor.

SELECT first\_name, last\_name

FROM sakila.actor;

1b. Display the first and last name of each actor in a single column in upper case letters. Name the column Actor Name.

SELECT UPPER(CONCAT(first\_name, ' ', last\_name)) AS Actor\_Name

FROM sakila.actor;

2a. You need to find the ID number, first name, and last name of an actor, of whom you know only the first name, "Joe." What is one query would you use to obtain this information?

SELECT actor\_id, first\_name, last\_name

FROM sakila.actor WHERE first\_name = "Joe"

2b. Find all actors whose last name contain the letters GEN:

SELECT \* FROM sakila.actor

WHERE last\_name LIKE '%GEN%';

2c. Find all actors whose last names contain the letters LI. This time, order the rows by last name and first name, in that order:

SELECT \*

FROM sakila.actor

WHERE last\_name LIKE '%LI%'

ORDER BY last\_name ASC, first\_name

2d. Using IN, display the country\_id and country columns of the following countries: Afghanistan, Bangladesh, and China:

SELECT country\_id, country

FROM sakila.country

WHERE country IN ('Afghanistan', 'Bangladesh', 'China');

3a. Add a middle\_name column to the table actor. Position it between first\_name and last\_name. Hint: you will need to specify the data type.

USE sakila;

ALTER TABLE actor ADD middle\_name VARCHAR(45) NOT NULL FIRST;

SELECT middle\_name, first\_name, last\_name FROM sakila.actor;

3b. You realize that some of these actors have tremendously long last names. Change the data type of the middle\_name column to blobs.

SELECT middle\_name, first\_name, last\_name FROM sakila.actor;

ALTER TABLE actor MODIFY COLUMN middle\_name Blob;

3c. Now delete the middle\_name column.

USE sakila;

SELECT middle\_name, first\_name, last\_name FROM sakila.actor;

ALTER TABLE actor DROP COLUMN middle\_name;

SELECT \* FROM sakila.actor;

4a. List the last names of actors, as well as how many actors have that last name.

USE sakila;

SELECT last\_name, COUNT(last\_name) as "Count of Last Names"

FROM sakila.actor

GROUP BY last\_name;

4b. List last names of actors and the number of actors who have that last name, but only for names that are shared by at least two actors

SELECT last\_name, COUNT(last\_name) as "Count of Last Names"

FROM sakila.actor

GROUP BY last\_name

HAVING COUNT(last\_name) >= 2

ORDER BY COUNT(last\_name) ASC;

4c. Oh, no! The actor HARPO WILLIAMS was accidentally entered in the actor table as GROUCHO WILLIAMS, the name of Harpo's second cousin's husband's yoga teacher. Write a query to fix the record.

—First I am making sure this is the correct record, I need to update

SELECT actor\_id, first\_name, last\_name

FROM sakila.actor

WHERE first\_name = 'GROUCHO'

and last\_name = 'WILLIAMS';

— now updating the record with correct info

UPDATE sakila.actor

SET first\_name = 'HARPO'

WHERE actor\_id = 172;

— Making sure it is updated correctly

SELECT actor\_id, first\_name, last\_name

FROM sakila.actor

WHERE actor\_id = 172;

4d. Perhaps we were too hasty in changing GROUCHO to HARPO. It turns out that GROUCHO was the correct name after all! In a single query, if the first name of the actor is currently HARPO, change it to GROUCHO. Otherwise, change the first name to MUCHO GROUCHO, as that is exactly what the actor will be with the grievous error. BE CAREFUL NOT TO CHANGE THE FIRST NAME OF EVERY ACTOR TO MUCHO GROUCHO, HOWEVER! (Hint: update the record using a unique identifier.)

UPDATE sakila.actor

SET first\_name = CASE

WHEN first\_name = 'HARPO' THEN 'GROUCHO'

ELSE 'MUCHO GROUCHO'

END

WHERE actor\_id = 172;

SELECT actor\_id, first\_name, last\_name

FROM sakila.actor

WHERE actor\_id = 172;

5a. You cannot locate the schema of the address table. Which query would you use to re-create it? HINT: CHECK THIS OUT

USE sakila;

DESC sakila.address; — to check all the datatypes

SHOW CREATE TABLE sakila.address;

-- CREATE TABLE `address` (

-- `address\_id` smallint(5) unsigned NOT NULL AUTO\_INCREMENT,

-- `address` varchar(50) NOT NULL,

-- `address2` varchar(50) DEFAULT NULL,

-- `district` varchar(20) NOT NULL,

-- `city\_id` smallint(5) unsigned NOT NULL,

-- `postal\_code` varchar(10) DEFAULT NULL,

-- `phone` varchar(20) NOT NULL,

-- `location` geometry NOT NULL,

-- `last\_update` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

-- PRIMARY KEY (`address\_id`),

-- KEY `idx\_fk\_city\_id` (`city\_id`),

-- SPATIAL KEY `idx\_location` (`location`),

-- CONSTRAINT `fk\_address\_city` FOREIGN KEY (`city\_id`) REFERENCES `city` (`city\_id`) ON UPDATE CASCADE

-- ) ENGINE=InnoDB AUTO\_INCREMENT=606 DEFAULT CHARSET=utf8

6a. Use JOIN to display the first and last names, as well as the address, of each staff member. Use the tables staff and address:

SELECT s.first\_name, s.last\_name, a.address

FROM address a

INNER JOIN staff s

WHERE a.address\_id = s.address\_id

6b. Use JOIN to display the total amount rung up by each staff member in August of 2005. Use tables staff and payment.

SELECT SUM(p.amount) as "Total Amount"

FROM staff s

INNER JOIN payment p

ON s.staff\_id = p.staff\_id

GROUP BY p.payment\_date LIKE '2005-08-%\_%\_';

SELECT first\_name, last\_name, SUM(amount) as "Total Amount"

FROM staff s

INNER JOIN payment p USING (staff\_id) -- equivalent to where s.staff\_id = p.staff\_id

GROUP BY p.staff\_id

ORDER by last\_name ASC

6c. List each film and the number of actors who are listed for that film. Use tables film\_actor and film. Use inner join.

— List each film and the number of actors who are listed for that film. Use tables film\_actor and film. Use inner join.

SELECT title, COUNT(a.actor\_id) as "Number of actors"

FROM film f

INNER JOIN film\_actor a

ON f.film\_id = a.film\_id

GROUP BY title;

6d. How many copies of the film Hunchback Impossible exist in the inventory system?

SELECT title, COUNT(inventory\_id) as "Number of film"

FROM film f

INNER JOIN inventory i

ON f.film\_id = i.film\_id

WHERE title = "Hunchback Impossible"

6e. Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the customers alphabetically by last name:

![Total amount paid](Images/total\_payment.png)

SELECT last\_name, first\_name, SUM(amount)

FROM payment p

INNER JOIN customer c

ON p.customer\_id = c.customer\_id

GROUP BY p.customer\_id

ORDER BY last\_name ASC;

7a. The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence, films starting with the letters K and Q have also soared in popularity. Use subqueries to display the titles of movies starting with the letters K and Q whose language is English.

select title

from film

where language\_id IN

(

select language\_id

from language

where name = 'English')

and (title like 'k%')

or (title like 'q%');

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ ls

README.md chromedriver scrape\_mars.py

Readme-Mission-to-Mars.docx chromedriver.exe style.css

\_\_pycache\_\_ mission\_to\_mars.ipynb table.html

app.py requirements.txt templates

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ git push origin master

Enter passphrase for key '/Users/sunitharamakrishnan/.ssh/id\_rsa':

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ ls

README.md chromedriver scrape\_mars.py

Readme-Mission-to-Mars.docx chromedriver.exe style.css

\_\_pycache\_\_ mission\_to\_mars.ipynb table.html

app.py requirements.txt templates

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ ls

README.md chromedriver scrape\_mars.py

Readme-Mission-to-Mars.docx chromedriver.exe style.css

\_\_pycache\_\_ mission\_to\_mars.ipynb table.html

app.py requirements.txt templates

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ pwd

/Users/sunitharamakrishnan/Documents/UCB\_HomeWork\_submission/HomeWork11-Web-Scraping

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ ls

README.md chromedriver scrape\_mars.py

Readme-Mission-to-Mars.docx chromedriver.exe style.css

\_\_pycache\_\_ mission\_to\_mars.ipynb table.html

app.py requirements.txt templates

Sunithas-MacBook-Pro:HomeWork11-Web-Scraping sunitharamakrishnan$ cd ../

Sunithas-MacBook-Pro:UCB\_HomeWork\_submission sunitharamakrishnan$ ls

Census+Business+Data.py HomeWork5-Matplotlib-old2 README.md

Census-Business-Data.ipynb HomeWork5-Matplotlib.old3 Untitled Folder

HomeWork1-Excel HomeWork6-Python-APIs Untitled Folder 1

HomeWork10-Web HomeWork7-social-analytics copyWeatherAPI.py.ipynb

HomeWork11-Web-Scraping HomeWork7-social-analytics.working data\_engineering-Copy1.ipynb

HomeWork11-practice HomeWork8-SQL sunitharamar.github.io

HomeWork2-NOT\_submitted\_huge\_file HomeWork9-Sequelize twitter.ipynb

HomeWork2-VBA HomeWork9-Sequelize.old twitter2.ipynb

HomeWork4\_Numpy\_Pandas Homework10-HTML working-Twitter-Social\_Analytics.ipynb

HomeWork5-Matplotlib JavaScript-and-DOM-Manipulation

HomeWork5-Matplotlib-old JavaScript-practice

Sunithas-MacBook-Pro:UCB\_HomeWork\_submission sunitharamakrishnan$ cd HomeWork8-SQL/

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ ls

README.md README.sql sakila\_schema.svg

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ clear

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ ls

README.md README.sql sakila\_schema.svg

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ pwd

/Users/sunitharamakrishnan/Documents/UCB\_HomeWork\_submission/HomeWork8-SQL

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ ls

README.md README.sql sakila\_schema.svg

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ vi README.md

Sunithas-MacBook-Pro:HomeWork8-SQL sunitharamakrishnan$ !!

vi README.md

FROM film f

INNER JOIN inventory i

ON f.film\_id = i.film\_id

WHERE title = "Hunchback Impossible"

6e. Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the customers alphabetically by last name:

![Total amount paid](Images/total\_payment.png)

SELECT last\_name, first\_name, SUM(amount)

FROM payment p

INNER JOIN customer c

ON p.customer\_id = c.customer\_id

GROUP BY p.customer\_id

ORDER BY last\_name ASC;

7a. The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence, films starting with the letters K and Q have also soared in popularity. Use subqueries to display the titles of movies starting with the letters K and Q whose language is English.

select title

from film

where language\_id IN

(

select language\_id

from language

where name = 'English')

and (title like 'k%')

or (title like 'q%');

7b. Use subqueries to display all actors who appear in the film Alone Trip.

select f.title, a.first\_name, a.last\_name

from film f, actor a, film\_actor fa

where f.film\_id = fa.film\_id

and fa.actor\_id = a.actor\_id

and f.title = "Alone Trip";

select first\_name, last\_name

from actor

where actor\_id in

(

select actor\_id

from film\_actor

where film\_id in

(

select film\_id

from film

where title = "Alone Trip"

)

);

7c. You want to run an email marketing campaign in Canada, for which you will need the names and email addresses of all Canadian customers. Use joins to retrieve this information.

select cu.first\_name, cu.last\_name, cu.email, a.address, ci.city, co.country

from customer cu

inner join address a

on cu.address\_id = a.address\_id

inner join city ci

on a.city\_id = ci.city\_id

inner join country co

on ci.country\_id = co.country\_id

where country in ('canada');

select cu.first\_name, cu.last\_name, cu.email, a.address, ci.city, co.country

from customer cu

right join address a

on cu.address\_id = a.address\_id

right join city ci

on a.city\_id = ci.city\_id

right join country co

on ci.country\_id = co.country\_id

where country = 'Canada';

select cu.first\_name, cu.last\_name, cu.email, a.address, ci.city, co.country

from customer cu

left join address a

on cu.address\_id = a.address\_id

left join city ci

on a.city\_id = ci.city\_id

left join country co

on ci.country\_id = co.country\_id

where country = 'Canada';

select first\_name, last\_name, email, address, city, country

from customer cu, country co, address a, city ci

WHERE cu.address\_id = a.address\_id

and a.city\_id = ci.city\_id

and ci.country\_id = co.country\_id

and country = 'Canada' ;

7d. Sales have been lagging among young families, and you wish to target all family movies for a promotion. Identify all movies categorized as famiy films.

select title, name

from film ,film\_category, category

where film.film\_id = film\_category.film\_id

and film\_category.category\_id = category.category\_id

and category.name = 'Family'

order by name DESC;

SELECT title, category

FROM film\_list

WHERE category = 'Family';

7e. Display the most frequently rented movies in descending order.

SELECT i.film\_id, f.title, COUNT(r.inventory\_id)

FROM inventory i

INNER JOIN rental r

ON i.inventory\_id = r.inventory\_id

INNER JOIN film\_text f

ON i.film\_id = f.film\_id

GROUP BY r.inventory\_id

ORDER BY COUNT(r.inventory\_id) DESC;

7f. Write a query to display how much business, in dollars, each store brought in.

select st.store\_id, concat('$',format(sum(amount),2)) as total\_business

from store st

join customer c

using (store\_id)

join payment p

on (c.customer\_id = p.customer\_id)

group by (st.store\_id);

7g. Write a query to display for each store its store ID, city, and country.

SELECT s.store\_id, st.first\_name as "Manager\_first\_name", st.last\_name as "Manager\_last\_name" , ci.city, co.country

FROM store s

INNER JOIN customer cu

ON s.store\_id = cu.store\_id

INNER JOIN staff st

ON s.store\_id = st.store\_id

INNER JOIN address a

ON cu.address\_id = a.address\_id

INNER JOIN city ci

ON a.city\_id = ci.city\_id

INNER JOIN country co

ON ci.country\_id = co.country\_id

7h. List the top five genres in gross revenue in descending order. (Hint: you may need to use the following tables: category, film\_category, inventory, payment, and rental.)

select ca.name, concat('$',format(sum(amount),2)) as "Revenue-Top\_5\_genres"

from category ca

inner join film\_category fc

on ca.category\_id = fc.category\_id

inner join inventory i

on fc.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

group by ca.name

order by SUM(p.amount) DESC

LIMIT 5;

8a. In your new role as an executive, you would like to have an easy way of viewing the Top five genres by gross revenue. Use the solution from the problem above to create a view. If you haven't solved 7h, you can substitute another query to create a view.

CREATE VIEW View\_Revenue\_Top\_5\_genres AS

select ca.name, concat('$',format(sum(amount),2)) as "Revenue-Top\_5\_genres"

from category ca

inner join film\_category fc

on ca.category\_id = fc.category\_id

inner join inventory i

on fc.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

group by ca.name

order by SUM(p.amount) DESC

LIMIT 5;

8b. How would you display the view that you created in 8a?

SELECT \* FROM View\_Revenue\_Top\_5\_genres

8c. You find that you no longer need the view top\_five\_genres. Write a query to delete it.

DROP VIEW View\_Revenue\_Top\_5\_genres

SELECT \* FROM View\_Revenue\_Top\_5\_genres

11:28:29 SELECT \* FROM View\_Revenue\_Top\_5\_genres LIMIT 0, 50000 Error Code: 1146. Table 'sakila.view\_revenue\_top\_5\_genres' doesn't exist 0.00033 sec

============================================================================