Faculty of Computing

CS220: Database Systems

Class: BESE-13A

Lab 07: SQL Joins

Date: November 30th, 2023

Time: 2:00 PM-5:00 PM

Instructor: Dr. Shah Khalid

Lab Engineer: Sundas Dawood

Name: Aimen Munawar Class: BESE-13-A CMS ID: 415867

Lab 07: SQL Joins

Introduction

A SQL JOIN clause combines records from two or more tables in a database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables by using values common to both tables.

Objectives

After performing this lab students should be able to:

- 1. Design SQL queries to retrieve data from multiple tables by using JOIN operation.
- 2. Select among INNER, OUTER, NATURAL, CROSS join as and when required.

Tools/Software Requirement

- MySQL Community Server 5.6 or later
- MySQL Workbench 6.1 or later
- Sakila Database and Company Database

Description

- 1. This lab assumes that MySQL Community Server is running, Sakila database has been loaded using MySQL Workbench, and the query window is open.
- 2. Execute the JOIN queries on Sakila including cross join, inner join, natural join, outer join (left outer join, right outer join, and full outer join) and observe how each join operation differs from the other.
- 3. For your practice, create the following two tables and execute the following queries to better understand how each join operation works.
 - a. Specify a primary-foreign key relationship between both tables.
 - b. Produce a set of all users who are enrolled in a course.
 - c. List of all students and their courses even if they're not enrolled in any course yet.
 - d. List all records of both tables regardless of any match.

Table: Users

id	name	course_id
1	Alice	1
2	Bob	1

CS220: Database Systems Page



3	Caroline	2
4	David	5
5	Emma	(NULL)

Table: Course

course_id	language
1	HTML5
2	NULL
3	JavaScript
4	PHP
5	MySQL

- 4. Try running the following SQL JOIN queries. You can try equivalent SQL queries for the same result as well.
 - a. List rental and return date for the movie BREAKING HOME (notice one of the join uses **on** clause whereas the other uses **using** clause. 13 entries of renting this movie should be found from the database.

use sakila;

select r.rental_date, return_date
from rental r join inventory i
using (inventory_id) join film f
on (f.film_id=i.film_id)
where f.title like 'BREAKING HOME';

b. List of movie titles that were never rented. There are 43 of them. Notice that there are two outer joins as the movie might be missing altogether from the inventory



National University of Sciences and Technology (NUST) School of Electrical Engineering and Computer Science but may still be included in the result set. On the other hand, movies might

be available in the inventory but never rented, the SQL query covers both cases.

select f.title
from film f left join inventory
i using (film_id) left join
rental r using (inventory_id)
where (i.inventory_id is null or r.rental_id is null);

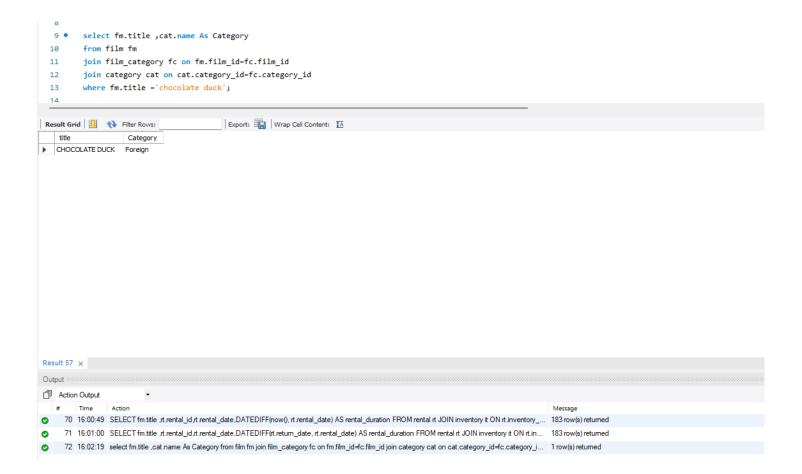
Lab Task

Write SQL queries for the following information needs. You should execute your attempts on SQL workbench and make necessary corrections if needed using **Sakila database**.

a. What category does the movie CHOCOLATE DUCK belong to?

Code:

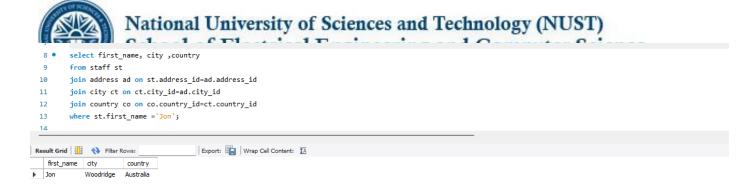
select fm.title ,cat.name As Category from film fm join film_category fc on fm.film_id=fc.film_id join category cat on cat.category_id=fc.category_id where fm.title ='chocolate duck';



b. Track the location (city & country) of staff member Jon.

Code:

select first_name, city ,country from staff st join address ad on st.address_id=ad.address_id join city ct on ct.city_id=ad.city_id join country co on co.country_id=ct.country_id where st.first_name ='Jon';

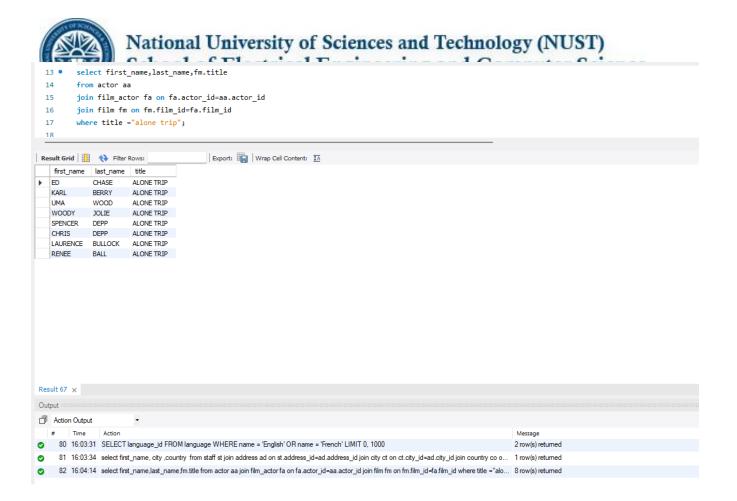




c. Retrieve first and last name of actors who played in ALONE TRIP.

Code:

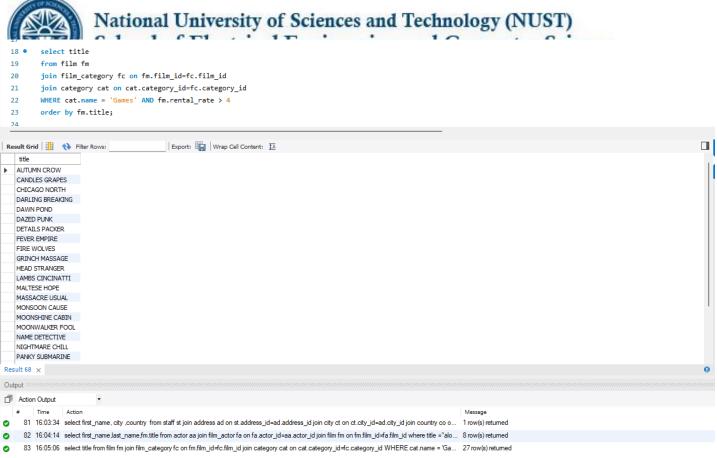
select first_name,last_name,fm.title from actor aa join film_actor fa on fa.actor_id=aa.actor_id join film fm on fm.film_id=fa.film_id where title ="alone trip";



d. List of movies in Games category having rental rate of more than \$4 and sorted on movie titles.

Code:

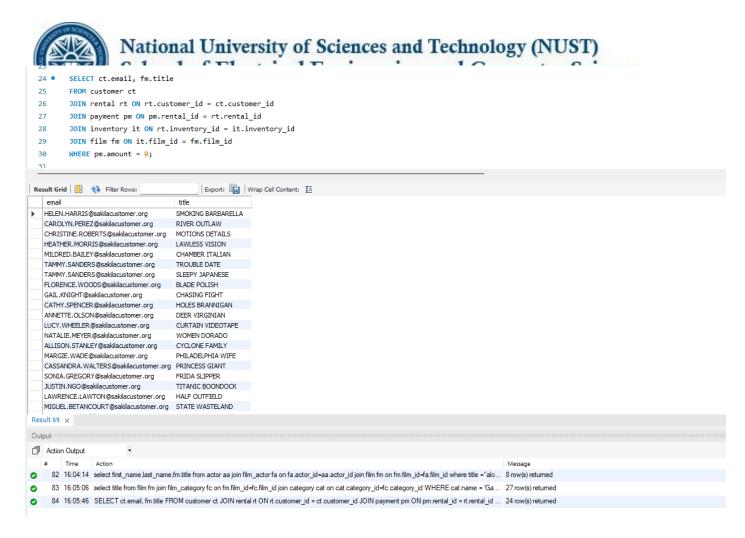
select title
from film fm
join film_category fc on fm.film_id=fc.film_id
join category cat on cat.category_id=fc.category_id
WHERE cat.name = 'Games' AND fm.rental_rate > 4
order by fm.title;



e. Email addresses of customers who rented a movie but didn't pay anything, include movie title in the output.

Code:

SELECT ct.email, fm.title
FROM customer ct
JOIN rental rt ON rt.customer_id = ct.customer_id
JOIN payment pm ON pm.rental_id = rt.rental_id
JOIN inventory it ON rt.inventory_id = it.inventory_id
JOIN film fm ON it.film_id = fm.film_id
WHERE pm.amount = 0;

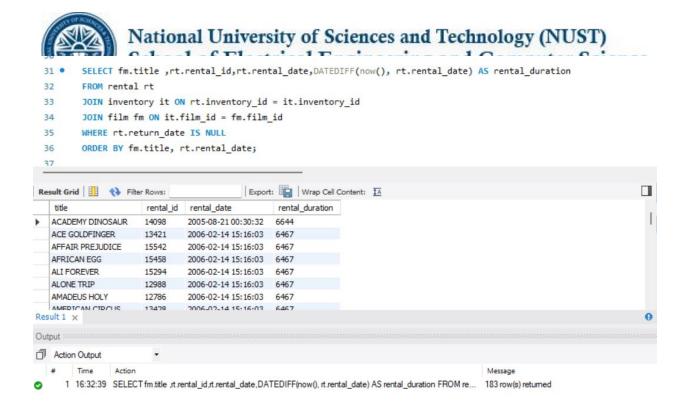


f. List of unpaid rentals (film title, rental id, rental date and for how many days rented). List should be sorted on film title and rental date.

Code:

Output:

SELECT fm.title ,rt.rental_id,rt.rental_date,DATEDIFF(now(), rt.rental_date) AS rental_duration
FROM rental rt
JOIN inventory it ON rt.inventory_id = it.inventory_id
JOIN film fm ON it.film_id = fm.film_id
WHERE rt.return_date IS NULL
ORDER BY fm.title, rt.rental_date;



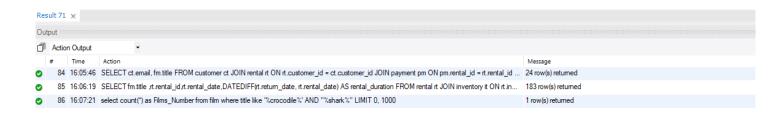
g. How many films involve the word "Crocodile" and a "Shark"?

Code:

```
select count(*) as Films_Number
from film
where title like '%crocodile%' AND "%shark%";
```

Output:

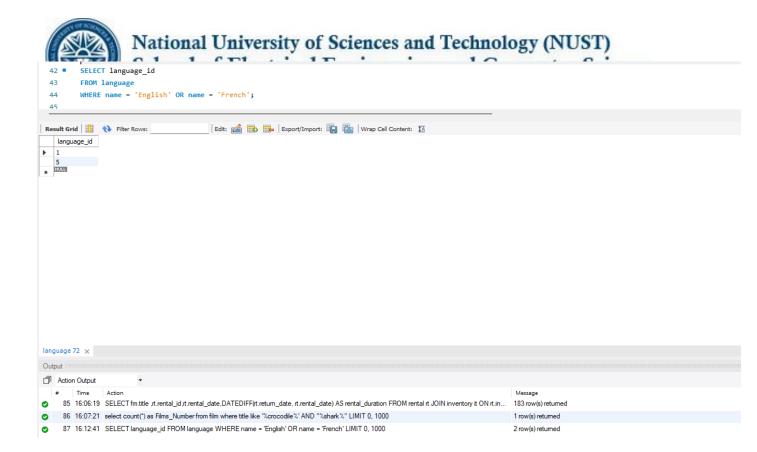




h. Retrieve language ids of languages ENGLISH and FRENCH.

Code:

SELECT language_id FROM language WHERE name = 'English' OR name = 'French'; Output:



Deliverable

Submit a PDF document including the SQL queries to answer above-mentioned information as well as snapshot of their outcome when executed over MySQL using the Workbench.