Lab 05 - Procedures, Functions, and Triggers

Due Feb 26 at 11:59pm **Points** 40 **Questions** 16 **Available** after Feb 24 at 10am **Time Limit** None
Allowed Attempts Unlimited

Instructions

In this lab we will focus mostly on advanced SQL topics like procedures, triggers, functions, etc. We will use a MySQL friendly subset of the Homework 2 IMDB dataset.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 3	3,051 minutes	37.05 out of 40
LATEST	Attempt 3	3,051 minutes	37.05 out of 40
	Attempt 2	87 minutes	0 out of 40 *
	Attempt 1	18 minutes	0 out of 40 *

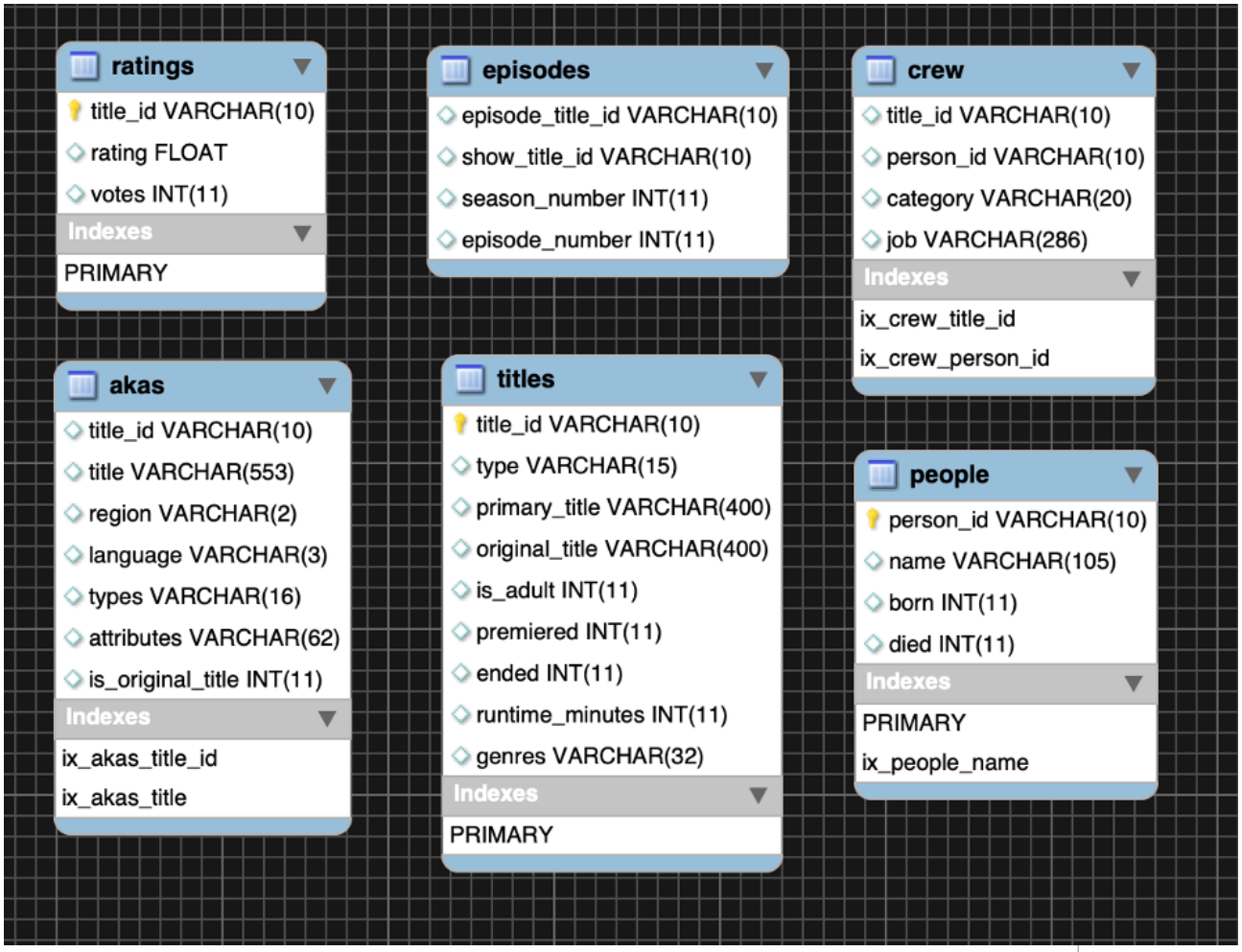
* Some questions not yet graded

Correct answers are hidden.

Score for this attempt: **37.05** out of 40
Submitted Feb 26 at 2:41pm
This attempt took 3,051 minutes.

The IMDB data is stored in: COMP_420_Spring_2021_Lab_05_imdb.sql

The table definitions are presented in the image below:



Due to the size of the initial dataset, the one you will work with contains only titles which had more than 5000 user ratings. Therefore, this dataset is a collection of the most popular titles on IMDB, there are 5786 titles in this set. The entities are as follows:

1. titles refers to individual distinct titles for a variety of types of media
2. akas refers to alternative titles
3. episodes refers to individual episodes of a tv show with a distinct title
4. people refers to distinct individuals
5. crew links people to titles through a job
6. ratings links titles to their user rating

To make querying faster, and to make it easier to load in the dataset, the database will not be enforcing referential integrity. That said, the following key relationships should be stable:

1. crew.person_id REFERENCES people.person_id
2. crew.title_id REFERENCES titles.title_id
3. ratings.title_id REFERENCES titles.title_id
4. akas.title_id REFERENCES titles.title_id
5. episodes.show_title_id REFERENCES titles.title_id
6. episodes.episode_title_id REFERENCES titles.title_id

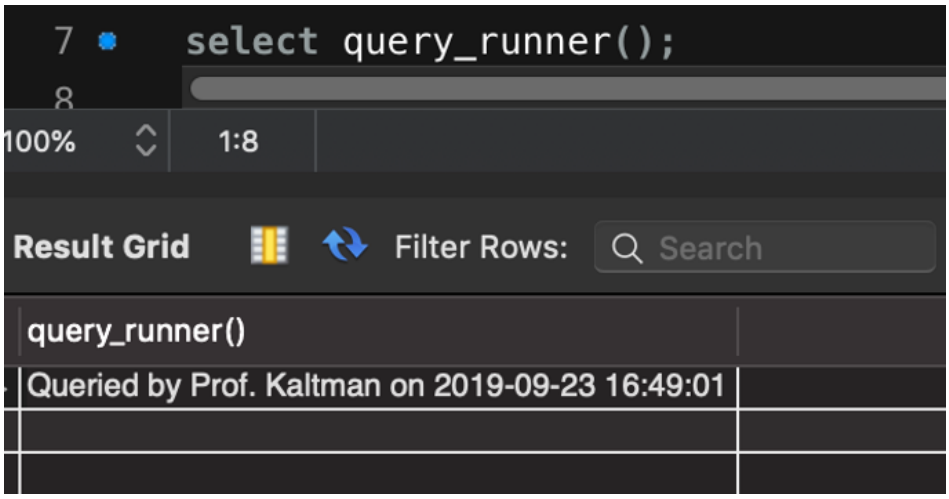
Question 1

4 / 4 pts

Part 1: From a View to a Query

This section of the lab will focus on views, sub- and correlated queries over the IMDB dataset.

For this first question, please write a user defined function (query_runner) that returns your name and the date of the query as in the picture below.



Insert query_runner's output into ALL of the following views and stored procedures.

Please upload a screenshot of your query and its output.

Lab5#1.JPG (https://cilearn.csuci.edu/files/2662581/download)

Question 2

3 / 3 pts

Construct a custom view (avg_by_performer) that lists actors and actresses and the average rating of their titles in the database in descending order.

Please upload a screenshot of your view declaration and its output.

```
26  select * from avg_by_performer;
27
28
```

100% 1:25

Result Grid Filter Rows: Search Export: Fetch rows:

Name	Average Rating	Query Runner / Date
Yoshimasa Hosoya	9.899999618530273	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Marina Inoue	9.899999618530273	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Kenshô Ono	9.899999618530273	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Yûto Uemura	9.899999618530273	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Tatsuhisa Suzuki	9.899999618530273	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Jo Wyatt	9.699999809265137	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Jaimi Barbakoff	9.699999809265137	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Rob Wiethoff	9.699999809265137	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Matthew Needham	9.699999809265137	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Robert Sean Leonard	9.650000095367432	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:
Radiohead	9.600000381469727	Queried by Prof. Eric Kaltman at 2019-09-24 11:39:

Lab5#2.JPG (https://cilearn.csuci.edu/files/2662583/download)

Question 3

3 / 3 pts

Construct a view (tv_seasons) consisting of the name of a television show and the number of seasons and episodes grouped by title then ordered by seasons then episodes.

Please upload a screenshot of your view declaration and its output.

```
15 •  select * from tv_seasons;
```

100% 4:13

Result Grid Filter Rows: Search Export:

Show Title	Num Seasons	Total Episodes	Query Runner / Date
Late Night with Jimmy Fallon	0	379	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Days of Our Lives	1	4180	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
EastEnders	1	2883	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Home and Away	1	2721	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Wan pîsu	1	357	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Teletubbies	1	142	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Yu Yu Hakusho: Ghost Files	1	42	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Mahabharat	1	36	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Monster	1	28	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
Samurai Champloo	1	12	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:
John Doe	1	11	Queried by Prof. Eric Kaltman at 2019-09-24 11:47:

Lab5#3.JPG (https://cilearn.csuci.edu/files/2662588/download)

Partial

Question 4

3.55 / 4 pts

Fill-in the blanks in the correlated sub-query below that shows the titles by each actor or actress that are greater than the average rating for all that performer's titles if they have more than 2 titles on the list. The query should generate the results posted below.

SELECT p1. [name] , category, [primary_title] , rating from people p1

JOIN crew USING(person_id)

JOIN titles USING([title_id])

JOIN [ratings] USING(title_id)

WHERE category = " [actor] " OR [category] = "actress"

AND [title_id] > (SELECT [avg] (rating) from people p2

JOIN crew USING(person_id)

JOIN ratings USING(title_id)

WHERE p1.person_id = [p2] . [person_id])

AND [title_id] IN ([select] [person_id] from people

JOIN [crew] USING(person_id)

JOIN titles USING(title_id)

group BY person_id

HAVING count (title_id) > 2)

ORDER BY rating desc ;

name	category	primary_title	rating
Yoshimasa Hosoya	actor	Midnight Sun	9.9
Yûki Kaji	actor	Midnight Sun	9.9
Yui Ishikawa	actress	Midnight Sun	9.9
Kenshô Ono	actor	Midnight Sun	9.9
Yûki Kaji	actor	That Day	9.9
Yûto Uemura	actor	That Day	9.9
Yui Ishikawa	actress	That Day	9.9
Tatsuhisa Suzuki	actor	That Day	9.9
Anna Gunn	actress	Face Off	9.9
Aaron Paul	actor	Face Off	9.9
Hugh Dancy	actor	Mizumono	9.9
Mads Mikkelsen	actor	Mizumono	9.9
Caroline Dhavernas	actress	Mizumono	9.9
Laurence Fishburne	actor	Mizumono	9.9
Peter Dinklage	actor	Battle of the Bastards	9.9
Peter Dinklage	actor	The Winds of Winter	9.9
Joel McHale	actor	Modern Warfare	9.8

Answer 1:

name

Answer 2:

primary_title

Answer 3:

title_id

Answer 4:

ratings

Answer 5:

actor

Answer 6:

category

Answer 7:

title_id

Answer 8:

AVG

Answer 9:

p2

Answer 10:

person_id

Answer 11:

title_id

Answer 12:

SELECT

Answer 13:

person_id

Answer 14:

crew

Answer 15:

GROUP

Answer 16:

count

Answer 17:

title_id

Answer 18:

desc

Part 2: Standard Operating Procedures (and Functions!)

Below you will implement some procedures, triggers and functions to provide more stability for the data in the IMDB data set.

Section 1: Procedural Thinking

In this section, please implement the following stored procedures and paste your procedure code, your call statement and the results below each answer. Make sure your input data types match the tables from which you are gathering information. ***Remember to include your 'query_runner' information within each procedure***.

Question 5

4 / 4 pts

Create a stored procedure (sp_alternate_titles) that takes a title and returns a listing of all its alternative titles (from the akas table) ordered alphabetically.

Please upload a screenshot of your stored procedure and the CALL returning the correct results.

```
12 call sp_alternate_titles('Frozen');
```

100% 1:11

Result Grid Filter Rows: Search Export:

title	query_runner()
????? ?? ???????	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
??????? ?????	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
???????? ??????	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
?????????	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
??????????? ????????	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
?adové krá?ovstvo	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Ana to Yuki no Joou	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Anna and the Snow Queen	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Bajo cero	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Beating the Mountain: Surviving Frozen	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Bingxue Qi Yuan	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Die Eiskönigin - Völlig unverfroren	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Donmak	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Frosinn	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32
Frost	Queried by Prof. Eric Kaltman at 2019-09-24 16:57:32

lab5#5.JPG (https://cilearn.csuci.edu/files/2662599/download)

Question 6

4 / 4 pts

Create a stored procedure (sp_worked_on) that takes a person's name and their category (from the crew table) and returns a list of the names of the titles they've worked on in alphabetical order.

Please upload a screenshot of your stored procedure and the CALL returning the correct results.

```
16 •  call sp_worked_on('Ridley Scott', 'director');
```

100% 1:15

Result Grid Filter Rows: Search Export:

primary_title	Query Runner / Date
A Good Year	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Alien	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Alien: Covenant	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Blade Runner	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Body of Lies	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Gladiator	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Legend	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Prometheus	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Robin Hood	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
The Counselor	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
The Duellists	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16
Thelma & Louise	Queried by Prof. Eric Kaltman at 2019-09-24 16:49:16

Lab5#6.JPG (https://cilearn.csuci.edu/files/2662600/download)

Section 2: Trigger Design and Execution

Please read through the next set of questions before starting this section of the lab. You will be setting up some basic DB triggers and then triggering them. Submissions will mainly be screenshots of each of the steps in this process. You will be modifying the IMDB tables and data in your MySQL instance. As a result, ***I would highly advise writing SQL that reverts the changes made in the following steps. You will need to test your trigger and that may require DELETEing and re-INSERTing records and ALTERing TABLEs to add and remove columns.***

Question 7

3 / 3 pts

Create a trigger (ins_movie_2021) that sets the ended value of an inserted movie to 0 and the premiered date to the current year (2021).

Please upload a screenshot of your trigger code from MySQL Workbench, and show the successful trigger creation message in the "Action Output" window.

lab5#7.JPG (https://cilearn.csuci.edu/files/2662601/download)

Question 8

1 / 1 pts

Construct a bulk insert statement to add two movies released in 2021 into the database. Enter the "premiered" date of each movie as some other year than 2021. This will allow us to test the trigger.

Please upload a screenshot of your insert statement and the successful insert message in the "Action Output" window.

lab5#8.JPG (https://cilearn.csuci.edu/files/2662602/download)

Question 9

1 / 1 pts

Run the following query:

```
SELECT * FROM titles WHERE premiered = 2021;
```

Please upload a screenshot of the query and its result. There should be two movies with the correct premiered date in the results.

Lab5#9.JPG (https://cilearn.csuci.edu/files/2662603/download)

Question 10

1.5 / 3 pts

Create a trigger (round_rating) that modifies inserted ratings records to round the votes down to 4975 if they are less than 4985 and up to 5000 if they are 4985 or greater.

Please upload a screenshot of your trigger code from MySQL Workbench, and show the successful trigger creation message in the "Action Output" window.

Lab5#10.JPG (https://cilearn.csuci.edu/files/2662604/download)

you are basing this on "rating" and not "votes"

Question 11

0 / 1 pts

Run the insert statements in COMP_420_Spring_2021_Lab_05_Almost_Ratings.sql to enter new data into the database.

Run the following query:

```
SELECT * FROM ratings WHERE ratings.votes < 5001;
```

Please upload a screenshot of the query and its result. The results should be modified by your trigger from the previous question.

Lab5#11.JPG (https://cilearn.csuci.edu/files/2662605/download)

Question 12

1 / 1 pts

Create a new table DUPLICATED_PEOPLE that copies the structure of the PEOPLE table.

Please upload a screenshot of your table creation statement and the successful creation message in the "Action Output" window.

Lab5#12.JPG (https://cilearn.csuci.edu/files/2662607/download)

Question 13

2 / 2 pts

Use the ALTER TABLE statement to add a “duplicate_insert_count” column to the PEOPLE table (not the DUPLICATED_PEOPLE table). Be sure to use the correct data type.

Please upload a screenshot that shows the new column is added to the PEOPLE table.

Lab5#13.JPG (https://cilearn.csuci.edu/files/2662609/download)

Question 14

2 / 2 pts

Update the PEOPLE table so that every person has a “duplicate_insert_count” of 0. (You might need to SET sql_safe_updates = 0 if you get a 1175 error).

Please upload a screenshot of your update statement and the successful update message in the "Action Output" window.

LAb5#14.JPG (https://cilearn.csuci.edu/files/2662610/download)

Question 15

3 / 3 pts

Create a trigger (people_upd) on the DUPLICATED_PEOPLE table that increases the “duplicate_insert_count” value in the PEOPLE table by one for every duplicate person inserted into the DUPLICATED_PEOPLE table.

For example, if you run:

```
INSERT INTO duplicated_people VALUES (‘nm0000134’, ‘Robert De Niro’, 1943);
```

then the statement:

```
SELECT * FROM people WHERE name = ‘Robert De Niro’;
```

should return the record:

(‘nm0000134’, ‘Robert De Niro’, 1943,null,1) where the ‘1’ is an increased duplicate_insert_count.

Do not overthink this trigger.

Please upload a screenshot of your trigger creation statement and the successful creation message from the "Action Output" window.

lab5#15.JPG (https://cilearn.csuci.edu/files/2662611/download)

Question 16

1 / 1 pts

Run the COMP_420_Spring_2021_Lab_05_Almost_People.sql file. Then run the query:

```
SELECT * FROM people WHERE duplicate_insert_count > 0;
```

Please upload a screenshot of the query and its result to enable for a check on the trigger implmentation.

lab5#16.JPG (https://cilearn.csuci.edu/files/2662612/download)

Quiz Score: **37.05** out of 40