

This update includes significant changes to the functionality of the script – see steps (4) through (9).

Introduction

This SQL script has been provided to you to assist in the development of your eventual Phase 3 submission, and Phase 4 submission if desired. Though we’ve referred to this script as an “autograder”, it’s much more helpful to think of it simply as an aid for your development and testing. It is NOT the comprehensive suite of tests that we will use for the final evaluation of your project on Demo Day – we will almost certainly add additional tests. Also, this script displays the differences between your results and the expected results, but it doesn’t display a final score. The exact scoring rubric will be determined later in accordance with the final testing script, but this is the basic intent for the general distribution of points/scores:

- [0 – 60] out of 100 points: Your system provides the correct answers for queries based on the initial dataset; no filtering required.
- [50 – 70] out of 100 points: Your system provides the correct answers for the queries based on the initial dataset with various combinations of filtered values.
- [60 – 90] out of 100 points: Your system provides the correct answers for the filtered and unfiltered queries, and also implements the required modification queries while maintaining data integrity.
- [80 – 100] out of 100 points: Your system supports filtered queries and modification queries correctly, and also implements the logical constraints indicated in the project requirements.

General Overview of the Testing Script

You can use this script by loading it into a tab/window in MySQL Workbench, and then executing the script just like we executed queries and exercises during the term. The script calls your stored procedures and loads the resulting data into a newly created table called **magic44_test_results**. It then compares the test results with the expected results stored in the **magic44_answers** table. Once the script has finished, you can use the **magic44_count_differences** and **magic44_content_differences** views to examine the differences between your results and the expected results for a correct system.

The **magic44_count_differences** view identifies test results that differ in terms of the number of rows being output, while the **magic44_content_differences** view (similar to the **magic44_scoring** table from earlier versions) identifies differences between the attribute values in the rows themselves.

- Empty **magic44_count_differences** and **magic44_content_differences** view results are ideal – it means that your system is producing the exact same answers that are expected by an otherwise correct system, with the exception of sorting (the current script doesn’t evaluate sorting correctness).
- On the other hand, if the program does display differences, then the most important thing is not to panic. This is an excellent opportunity to examine the differences, and then analyze your system to determine which aspects need to be changed to correct the issues. These troubleshooting skills can be very valuable during the demo session as well, and for your future problem-solving challenges in general.

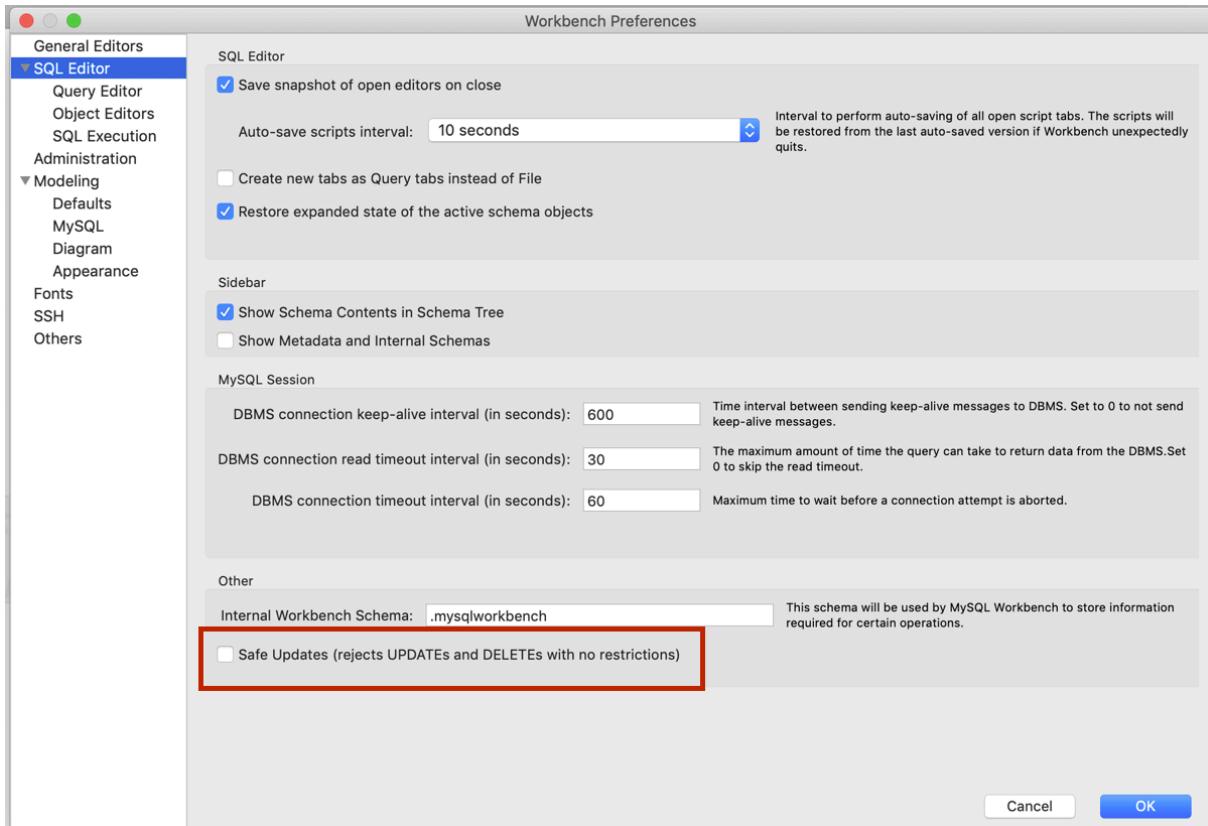
Remember that this script doesn’t include all of the tests that will be used during the demo session, so you should test those other aspects of the system on your own as thoroughly as possible.

Recommendations for Using the Script (and Development & Testing in General):

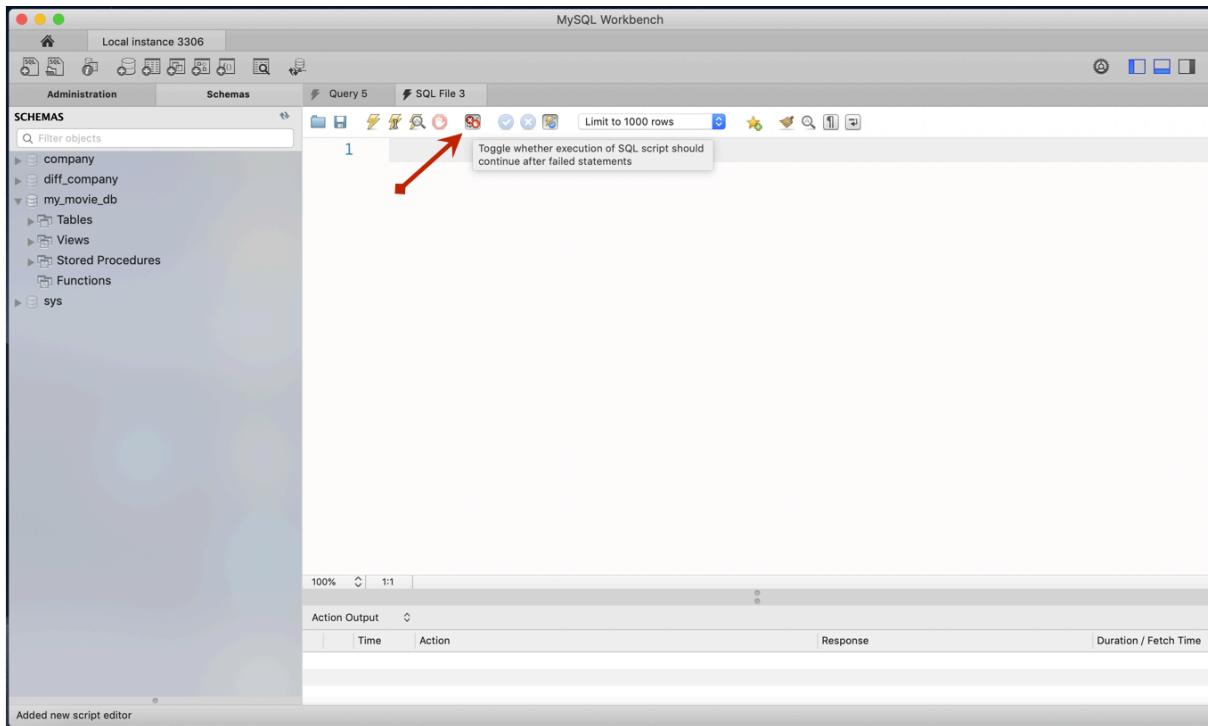
- Consider developing a `reset_initial_data()` stored procedure that deletes the current data from your tables and then reloads the initial dataset. This is very helpful when developing your modification queries, and saves time as opposed to having to reload a copy of your "initial state database" from a file.
- If you are allowing the testing script to execute fully and produce multiple errors, then consider focusing on the "earlier" testing errors first. Achieving success on earlier tests (e.g., showing that the basic table structures are solid and that the initial data is loaded correctly) is essential to demonstrating that the more advanced functionality (e.g., modification queries and handling of logical constraints) is also correct.
- You are allowed to create other database structures to support your stored procedures. One example includes creating views. Some of the queries that you'll create may seem relatively large and very complicated, involving the nesting of two or more levels of queries. One way to ease the complexity is to develop the "nested query portions" of the larger queries as separate views. If a query is saved as a view, then you can reference that view inside a larger query (i.e., "virtual table") as if it were a real table. Good examples include using the view as "source data" as part of the FROM clause, or to generate a list of values as part of an IN or a NOT IN clause.
- The Action Output section below the Result Grid might contain useful information for troubleshooting. If you encounter one or more errors during the testing of your script, then there's a good chance that one or more of your stored procedures simply didn't execute correctly to completion. You should work to ensure that your stored procedures execute correctly and consistently before being concerned with the numbers of rows and row content.

How to Use the Testing Script

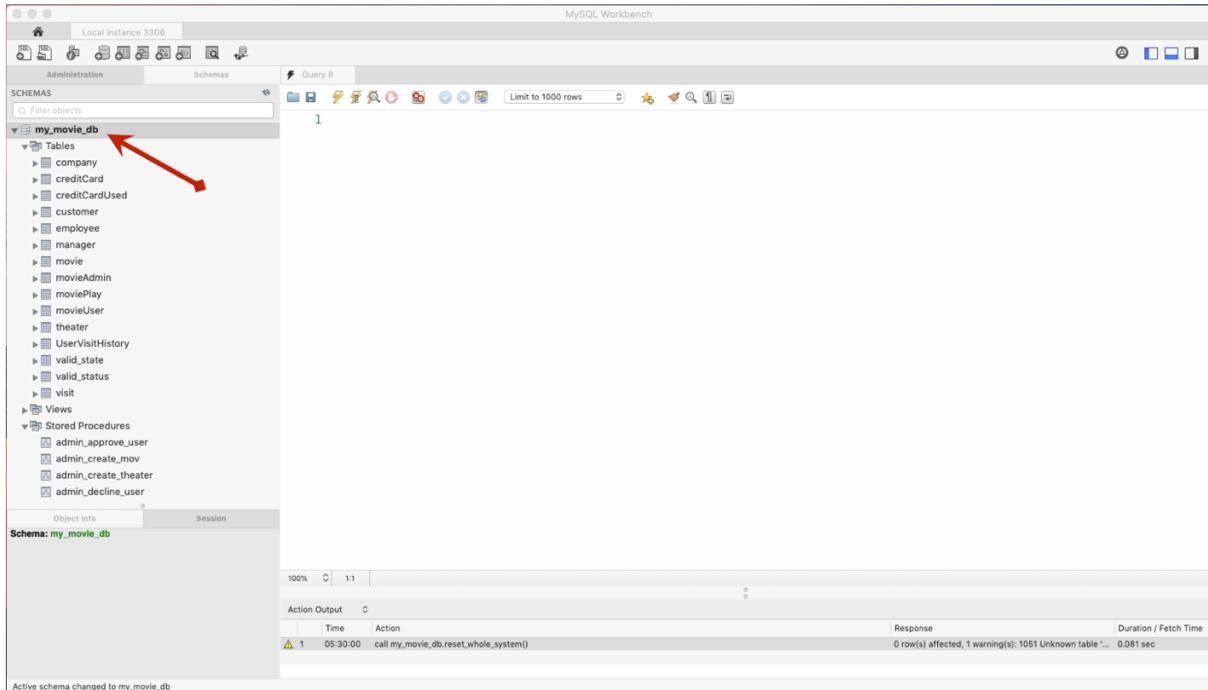
(1) Deactivate the "Safe Update" feature in the SQL Editors in MySQL Workbench. This will prevent the MySQL Workbench system from "interfering" with your (possibly unrestricted) modification queries.



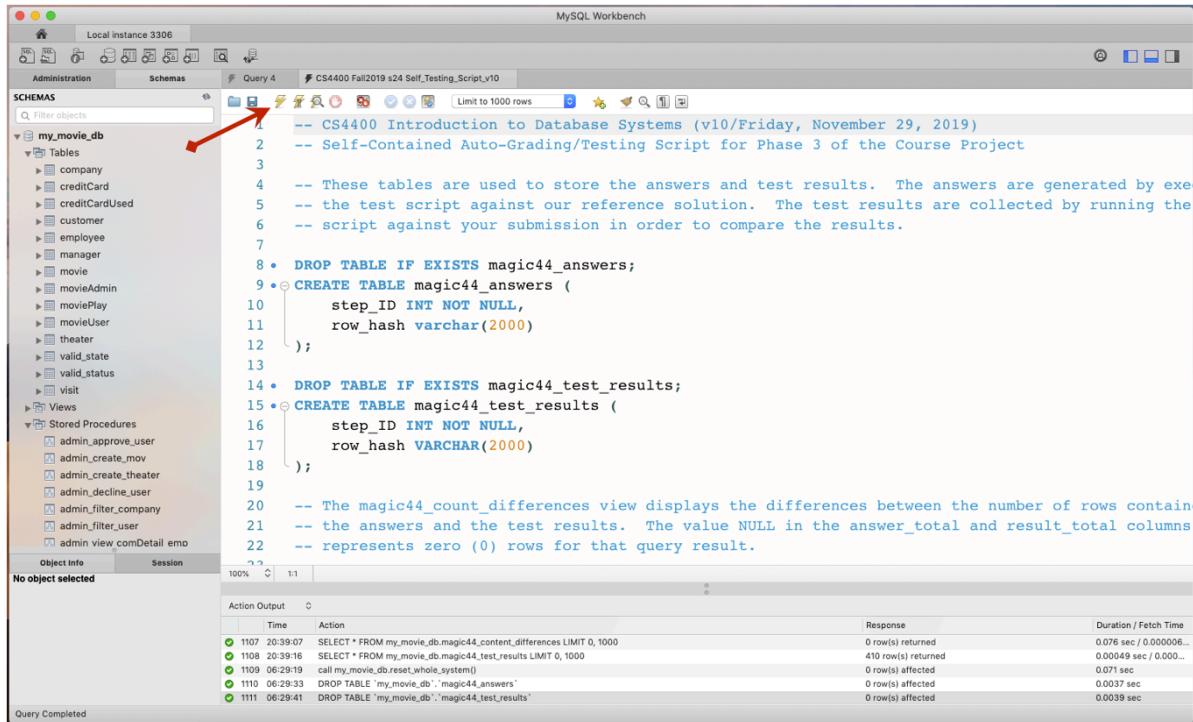
(2) Take a look at the “Continue Script Execution on Error” switch in MySQL Workbench. In early runs, you might prefer to have the script stop immediately when the first error is encountered. In later runs, though, you might prefer to continue running and examine all of the errors once the script has finished.



(3) Make sure that your database: (a) is loaded and highlighted; and, (b) contains the initial dataset. If your database does not contain the initial dataset, then it will likely fail most – if not all – of the test cases. Also, "**my_movie_db**" is just the name for the database in this instructional example. Make sure that your database uses the correct team naming conventions for your database per the Phase 3 Instructions. The name of the database should not impact the execution of this script, but it might impact some of our other testing systems.



(4) You can either: (a) select "RUN SQL Script" to execute the tests directly from the file; or, (b) you can select "Open SQL Script" to load the file contents into a new tab/window (as shown below) and execute the tests.



```

MySQL Workbench
Local instance 3306
Administration Schemas Query 4 CS4400 Fall2019 s24 Self_Testing_Script_v10
Limit to 1000 rows
-- CS4400 Introduction to Database Systems (v10/Friday, November 29, 2019)
-- Self-Contained Auto-Grading/Testing Script for Phase 3 of the Course Project
-- These tables are used to store the answers and test results. The answers are generated by executing
-- the test script against our reference solution. The test results are collected by running the
-- script against your submission in order to compare the results.
8 • DROP TABLE IF EXISTS magic44_answers;
9 • CREATE TABLE magic44_answers (
10     step_ID INT NOT NULL,
11     row_hash varchar(2000)
12 );
13
14 • DROP TABLE IF EXISTS magic44_test_results;
15 • CREATE TABLE magic44_test_results (
16     step_ID INT NOT NULL,
17     row_hash VARCHAR(2000)
18 );
19
20 -- The magic44_count_differences view displays the differences between the number of rows contained
21 -- in the answers and the test results. The value NULL in the answer_total and result_total columns
22 -- represents zero (0) rows for that query result.

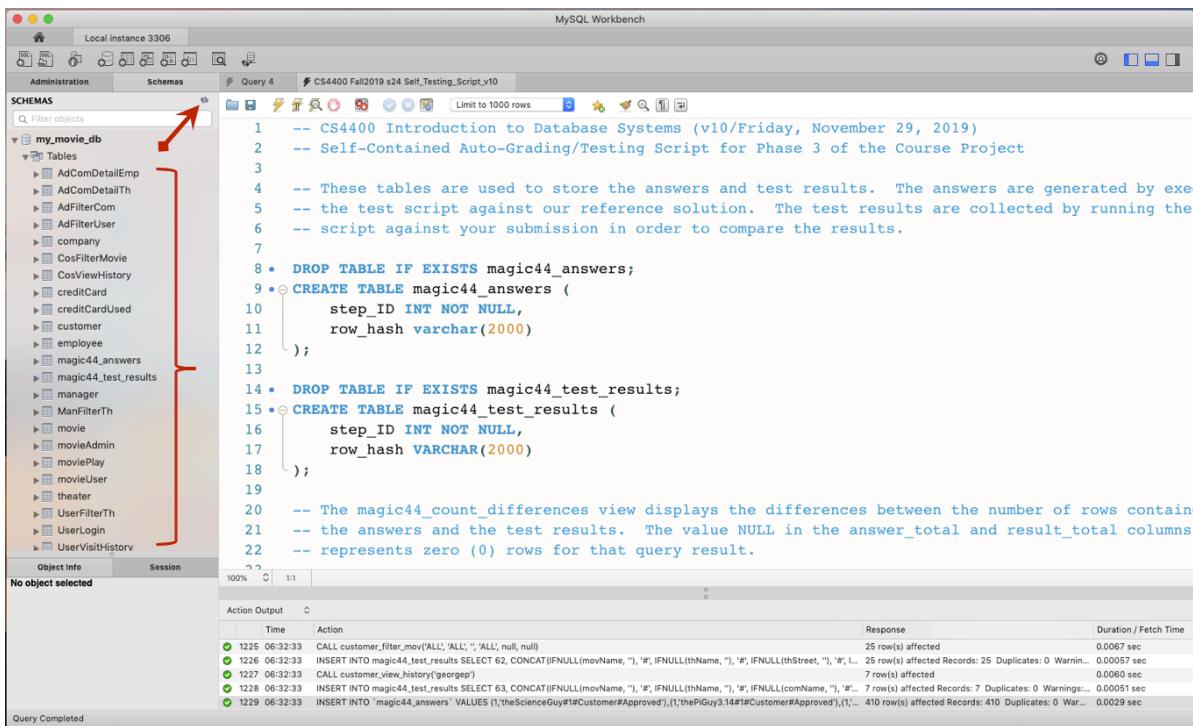
Object Info Session
No object selected
100% 1:1
Action Output
Time Action Response Duration / Fetch Time
1107 20:39:07 SELECT * FROM my_movie.db.magic44_content_differences LIMIT 0, 1000 0 row(s) returned 0.076 sec / 0.000006...
1108 20:39:16 SELECT * FROM my_movie.db.magic44_test_results LIMIT 0, 1000 410 row(s) returned 0.00049 sec / 0.000...
1109 06:29:19 call my_movie.db.reset_whole_system() 0 row(s) affected 0.071 sec
1110 06:29:33 DROP TABLE 'my_movie.db'.`magic44_answers` 0 row(s) affected 0.0037 sec
1111 06:29:41 DROP TABLE 'my_movie.db'.`magic44_test_results` 0 row(s) affected 0.0039 sec

Query Completed

```

If your program encounters errors, then the testing script might not have been fully completed (depending on your "Continue Script Execution" settings). Check the Action Output section [located near the bottom of the MySQL Workbench window] for the error(s), make corrections as needed, and try again.

(5) Be sure to refresh your view of the tables after the script has stopped. Note that there should be some extra tables present, including the **magic44_answers** and **magic44_test_results** tables that contain the current and expected testing results.



```

MySQL Workbench
Local instance 3306
Administration Schemas Query 4 CS4400 Fall2019 s24 Self_Testing_Script_v10
Limit to 1000 rows
-- CS4400 Introduction to Database Systems (v10/Friday, November 29, 2019)
-- Self-Contained Auto-Grading/Testing Script for Phase 3 of the Course Project
-- These tables are used to store the answers and test results. The answers are generated by executing
-- the test script against our reference solution. The test results are collected by running the
-- script against your submission in order to compare the results.
8 • DROP TABLE IF EXISTS magic44_answers;
9 • CREATE TABLE magic44_answers (
10     step_ID INT NOT NULL,
11     row_hash varchar(2000)
12 );
13
14 • DROP TABLE IF EXISTS magic44_test_results;
15 • CREATE TABLE magic44_test_results (
16     step_ID INT NOT NULL,
17     row_hash VARCHAR(2000)
18 );
19
20 -- The magic44_count_differences view displays the differences between the number of rows contained
21 -- in the answers and the test results. The value NULL in the answer_total and result_total columns
22 -- represents zero (0) rows for that query result.

Object Info Session
No object selected
100% 1:1
Action Output
Time Action Response Duration / Fetch Time
1225 06:32:33 CALL customer.Filter_mov('ALL', 'ALL', null, null) 25 row(s) affected 0.0067 sec
1226 06:32:33 INSERT INTO magic44_test_results SELECT 62, CONCAT(IFNULL(movName, ''), '#', IFNULL(thName, ''), '#', IFNULL(thStreet, ''), '#', IFNULL(viewName, ''), '#', IFNULL(historyName, '')) 25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0 0.00057 sec
1227 06:32:33 CALL customer.view.history('george')
1228 06:32:33 INSERT INTO magic44_test_results SELECT 63, CONCAT(IFNULL(movName, ''), '#', IFNULL(thName, ''), '#', IFNULL(thStreet, ''), '#', IFNULL(viewName, ''), '#', IFNULL(historyName, '')) 7 row(s) affected Records: 7 Duplicates: 0 Warnings: 0 0.00051 sec
1229 06:32:33 INSERT INTO `magic44_answers` VALUES (1,'theScienceGuy#1#Customer#Approved'),(1,'thePGuy3.14#1#Customer#Approved'),(1,... 410 row(s) affected Records: 410 Duplicates: 0 Warnings: 0 0.0029 sec

Query Completed

```

(6) The tables contain the raw data, and the script also generates four views: two primary views and two supporting views. The **magic44_count_differences** and **magic44_content_differences** views provide information about how your data compares to the test results. An empty table means that no significant differences (perhaps besides sort order) were detected.

(7) The **magic44_count_differences** view provides information about the number of rows in your test results compared to the number of expected number of rows. A **NULL** value in the answer_total or result_total columns means "zero rows" – effectively that the empty table was produced as the answer. An empty table is not necessarily incorrect, but if all of your queries are producing empty tables, then there might be a deeper problem with your query.

The screenshot shows the MySQL Workbench interface with the following details:

- Top Bar:** MySQL Workbench, Local Instance 3306.
- Schemas:** my_movie_db selected.
- Tables:** magic44_content_differences, magic44_count_answers, magic44_count_differences, magic44_count_test_results, manage_company, manage_company_all, manage_company_count_cities, manage_company_count_empl..., manage_company_count_theat..., manage_company_num_cities..., manage_company_num_emplo..., manage_company_num_theat..., manage_credit_card_count, manage_credit_card_count_on..., manage_credit_card_count_zero, manage_customer_filter, manage_customer_shows, manage_shows_capacity, manage_shows_filled, manage_shows_now_playing, manage_theater_free_managers, manage_theater_not_played.
- Query Editor:** Query 4, CS4400 Fall2019 s24 Self_Testing_Script_v10, magic44_count_differences. The query is: `SELECT * FROM my_movie_db.magic44_count_differences;`
- Result Grid:** Shows the results of the query:

step_ID	answer_total	result_total
8	26	30
9	3	30
10	1	30
28	1	NULL
29	1	NULL
30	1	NULL

- Action Output:** Shows the execution log:

Time	Action	Response	Duration / Fetch Time
06-44-21	INSERT INTO magic44_test_results SELECT 62, CONCAT(IFNULL(movieName, "#"), IFNULL(thName, "#"), IFNULL(thStreet, "#"), ...)	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.00054 sec
06-44-21	CALL customer_view_history('george')	7 row(s) affected	0.0056 sec
1351 06-44-21	INSERT INTO magic44_test_results SELECT 63, CONCAT(IFNULL(movieName, "#"), IFNULL(thName, "#"), IFNULL(comName, "#"), ...)	7 row(s) affected Records: 7 Duplicates: 0 Warnings: 0	0.00042 sec
1352 06-44-21	INSERT INTO `magic44_answers` VALUES ('theScienceGuy#Customer#Approved'),(l_therPiGuy3.14#Customer#Approved),(...)	410 row(s) affected Records: 410 Duplicates: 0 Warnings: 0	0.00030 sec
1353 06-44-49	SELECT * FROM my_movie_db.magic44_count_differences LIMIT 0, 1000	6 row(s) returned	0.0014 sec / 0.00000 sec

- Status Bar:** Query Completed.

(8) The **magic44_content_differences** view provides information about the attribute values in each row of the results tables compared to the number of expected number of rows. The attribute values of the row are "hashed" into a single string separated by the pound sign delimiter ('#'). Any **NULLs** in the attribute values are replaced by blank strings or zeros in the row hash depending on the resulting data type.

The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected, displaying the 'my_movie_db' schema. In the center, the 'Query Grid' shows the results of the query: 'SELECT * FROM my_movie_db.magic44_content_differences;'. The results are listed in a table with columns: step_ID, category, and row_hash. The data includes various entries such as 'extra', 'missing', and 'clarinetbeast#0#Customer#Declined'. The bottom section of the interface shows the 'Action Output' pane, which displays the execution history of the script, including various SQL statements and their responses.

step_ID	category	row_hash
10	extra	theScienceGuy#1#Customer#Approved
10	extra	radioactivePoRa#0#Manager#Approved
10	extra	isthisthekrustykrab#3#Customer#Approved
10	extra	eeqmcsquare#1#Customer#Approved
10	extra	fatherAI#0#Manager#Approved
10	extra	clarinetbeast#0#Customer#Declined
28	missing	calcwizard#Approved#1#0#0
29	missing	gdanger#Declined#0#0#0
30	missing	imbatman#Approved#0#0#1

(9) Finally, you can scroll down the testing script to see the actual calls that are being made to your stored procedures.

The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected, displaying the 'my_movie_db' schema. In the center, the 'Query Grid' shows the results of the query: 'SELECT * FROM my_movie_db.magic44_content_differences;'. The results are listed in a table with columns: step_ID, category, and row_hash. The data includes various entries such as 'extra', 'missing', and 'clarinetbeast#0#Customer#Declined'. The bottom section of the interface shows the 'Action Output' pane, which displays the execution history of the script, including various SQL statements and their responses. Three red arrows point to specific lines in the script: line 86, line 88, and line 92, which correspond to the 'magic44_test_results' insert statements.

step_ID	category	row_hash
10	extra	theScienceGuy#1#Customer#Approved
10	extra	radioactivePoRa#0#Manager#Approved
10	extra	isthisthekrustykrab#3#Customer#Approved
10	extra	eeqmcsquare#1#Customer#Approved
10	extra	fatherAI#0#Manager#Approved
10	extra	clarinetbeast#0#Customer#Declined
28	missing	calcwizard#Approved#1#0#0
29	missing	gdanger#Declined#0#0#0
30	missing	imbatman#Approved#0#0#1

Good luck as you finish the development and testing of your system.