|  |  |
| --- | --- |
| -  Sindy Saintclair  Tuesday, January 18, 2022  Lesson 3 – CRUD in SQL | |
| **Learning Objectives and Questions** | **Notes and Answers** |
| **INTRODUCTION** | During this lesson, you are going to explore CRUD operations in SQL commands. You may be wondering, what is CRUD? *CRUD* stands for Create, Read, Update, and Delete. This is a term you will run into during your career, so it is important to understand it. When executing CRUD in SQL, you will be using slightly different keyword. Below are the SQL commands that are equivalent to CRUD:  CRUD name SQL command  Create Insert  Read Select  Update Update  Delete Delete  As you can see, Update and Delete use the same keywords. You have already explored SELECT in past lessons, so you will now examine the rest of CRUD operations in SQL. |
| **INSERT INTO** | The INSERT INTO statement is used to insert data into a table within a database. There are two ways to insert data into the database. The first way is to specify both the columns and the values of the columns:  INSERT INTO table\_name (column1, column2, column3...)  VALUES (value1, value2, value3...);  The second way is to insert values for every column in the table. If this is the case, you do not need to specify the column names, just the values:  INSERT INTO table\_name  VALUES (value1, value2, value3...);  When inserting data into every existing column, make sure the order of the values is the same as the order of the columns.  Click “Next” to see more Insert examples. |
| **INSERT Examples** | Below are more Insert examples. Go ahead and get started!  *Example 1: Specify Columns and Values*  Consider the following query:  insert into sakila.actor (first\_name, last\_name, last\_update)  values ("Johnny", "Smith", "2019-01-17 08:43:11");  The above query may look like a lot, but all you are doing is inserting a new actor into the table actor.  If you want to see the actor in the table, you can run the following query:  SELECT \* FROM sakila.actor;  If you scroll down, Johnny Smith is the last actor listed. Did you notice that you didn’t specify the actor\_id, but it gave a value of 201? That is because it is an auto-increment field that will automatically generate an id when a new record is inserted.  If you wanted only to see Johnny Smith’s information, you could add a WHERE statement:  select \* from sakila.actor  where last\_name = "Smith";  The above query will pull in just the data of the customer with the last-name of Smith.  *Example 2: Specify Just Values*  Now, instead of what you did above, you could use the second way of inserting to save time writing out every column name:  insert into sakila.actor  values ("Kermit", "DaFrog", "2019-01-19 08:56:12");  If you run the above query, you will get an error:  Error Code: 1136. Column count doesn't match value count at row 1  As you can see, MySQL is complaining because you only have 12 values when there are 13 columns. Can you guess what the missing column is? It’s actor\_id! In the previous query, you listed out every column that data was being inserted to and actor\_id was automatically generated. But in this second insert into, you will have to define the actor\_id because MySQL is assuming you are inserting and defining data for EVERY column. You would have to run the below query:  insert into sakila.actor  values (202, "Kermit", "DaFrog", "2019-01-19 08:56:12");  The above query will insert the new actor without any problems.  *Example 3: Insert Data when Some is Missing*  Sometimes, you want to insert a new row, but you don’t have all the data for each column. If that is the case, there is an automatic fix for that. Consider below:  insert into sakila.actor (first\_name, last\_name)  values ("Miss", "Pigee");  Above, you only have the information for two columns. If you run a query to look up this new actor, you will see the following output:  Select Sarah. Output from a query. It says Select from new customers where first name equals Sarah.  Figure 3-4; Select Miss Pigee  Great work! Now that you feel good about inserting data into a column, you will explore NULL. |
| **What is NULL?** | A colun field that does not have a value will be read as a Null value. If a column allows for optional data, it is possible to insert a new records with no value. If this happens, then the record will be saved with a Null value. It is essential to understand that a Null value does not mean a value of zero or a field that contains spaces as its value. *Null* means that the value was left blank when the record was created.  It is possible to check using SQL keywords for Null values or non-Null values. You’ll use the IS NULL and IS NOT NULL keywords in MySQL. Below is the syntax for NULL and IS NOT NULL.  NULL:  SELECT column\_names FROM table\_name  WHERE column\_name IS NULL;  IS NOT NULL:  SELECT column\_names FROM table\_name  WHERE column\_name IS NOT NULL;  Now you will try it out!  *NULL*  If you want to check for values in a particular table that are Null, you can use the IS NULL keyword. You need this keyword because you are unable to check for Null values using operators such as =, >, etc. about which you have previously learned. For example, you could not run the below query:  select \* from address  where address2 = Null;  That will give you no results. But if you run the following query, you will see all customers that do not have a company attached to them:  select \* from address  where address2 is null;  And here is you MySQL output.  1 47 MySakila Drive Alberta 300 ... 2014-09-25 22:30:27  2 28 MySQL Boulevard QLD 576 ... 2014-09-25 22:30:09  3 23 Workhaven Lane Alberta 300 14033335568 ... 2014-09-25 22:30:27  4 1411 Lillydale Drive QLD 576 6172235589 ... 2014-09-25 22:30:09  ```  As you can see, you selected the `address2` column from the `address` table and pulled in all actors that have a `address2` field with the value of `Null`, and you get a total of 5 rows.  ---  ## IS NOT NULL  Now, if you want to check for data that is not null, you could flip around what you did in the `IS NULL` example:  ```sql  select \* from address  where address2 is not null;  As you can see, the query is pulling in all actors that have non-Null data for the address2 column, which is the preponderance of people – 605! What if you want to pull up everything that has multiple null values? At first, you might want to something like this using the AND keyword:  select \* from staff  where picture and password is null;  But the above query will return nothing. If you look in the staff table, there is clearly an entry there fulfilling the criteria! Well, using the AND keyword is correct, but you need to check if each column is not null separately, like below:  select \* from staff  where picture and password is null;  The output of above will be this:  2 Jon Stephens 4  Jon.Stephens@sakilastaff.com 2 1 Jon  2006-02-15 03:57:16  You could also do something like:  select \* from staff  where picture is null and store\_id = 2;  You’ll come up with the same result!  You have learned so much about SQL already; you can start putting what you have learned together! Feel free to try different queries and practice what you can do so far! |
| **UPDATE** | You have now explored the first two operations in CRUD: Create and Read, which in SQL are referred to as Insert and Select. You will now learn about updating your database!  *Update a Table*  Now that you feel comfortable with Insert and Select, Update will be fairly straightforward. When updating, you have to use a new keyword: SET. This will set a the field you want to be update to the new data.  Below is the syntax for Update:  UPDATE table\_name  SET column1 = value1, column2 = value2, ...  WHERE condition;  Go ahead and update the actor Johnny Smith that you inserted into the database earlier:  update actor  set first\_name = "Jonathan"  where actor\_id = 201;  Once the query is executed successfully, go ahead and view that specific customer using the below query:  select \* from actor  where actor\_id = 201;  As you can see from the output, this customer’s first name is now “Jonathan” instead of “Johnny”.  *Update Multiple Records*  What if you wanted to update a lot of things at once? Perhaps you want to set every customer in a particular store to active. You can do that all in one query!  update customer  set active = 1  where store\_id = 2;  And if you look at all customers within the second store, you now see their store\_id is 2:  *Warning!*  If you don’t include the WHERE statement when updating data, it will update every row to what you have SET.  Consider below:  update customer  set active = 1;  Without a WHERE statement, all customers will be updated to be active. Be careful, because if this is not your goal, you could update large amounts of data that shouldn’t be updated! |
| **DELETE** | *Delete* is a way to remove existing data from a table. Below is the syntax for deleting:  DELETE FROM table\_name  WHERE condition;  Well, Jonathan (Johnny) is now no longer an actor, and you’ve been asked to delete him from the actor table. Below is the query to delte Johnny from the table:  delete from actor  where actor\_id = 201;  If you run a select query to see an actor with an actor\_id of 201, you will see that no results are found. Try it! |