Ironhack Student Portal

preview.my.ironhack.com/lms/courses/course-

Lesson Goals

- Learn about the different types of action queries in MySQL.
- Learn about how to create new tables with make table queries.
- Learn how to remove records from tables with delete queries.
- Learn how to add new records to tables with append queries.
- Learn how to change and update table values with update queries.

Introduction

Up until this point, we have been using `SELECT` queries to retrieve data from a database. However, this is only part of what queries can help you do. There is an entire class of queries called *action queries* that let you perform various actions such as:

- Creating a new table
- Appending records into a table
- Deleting records from a table
- Updating records in a table

In this lesson, we will learn about these action queries and how to append, delete, and update records using them in MySQL.

Make Table Queries

A *make table query* creates a new, permanent table in the database from the results of a query. This is similar to how a temp table is created, but the table resulting from a make table query will still exist in the database after the SQL session has ended whereas a temp table will not.

For example, if we wanted to create a permanent table containing the total number of orders, items sold, and quantity sold per store in our publications database, we would do so using the `CREATE TABLE` command as follows.

```
CREATE TABLE publications.store_sales_summary
SELECT stores.stor_id AS StoreID, stores.stor_name AS Store, COUNT(DISTINCT(ord_num)) AS Orders,
COUNT(title_id) AS Items, SUM(qty) AS Qty
FROM publications.sales sales
INNER JOIN publications.stores stores ON stores.stor_id = sales.stor_id
GROUP BY StoreID, Store;
```

Even if we end our SQL session and start a new one, we will still be able to query this table as it is now a permanent table in the database.

```
SELECT *
FROM publications.store_sales_summary;
```

StoreID	Store	Orders	Items	Qty
6380	Eric the Read Books	2	2	8
7066	Barnum's	2	2	125
7067	News & Brews	2	4	90
7131	Doc-U-Mat: Quality Laundry and Books	3	6	130
7896	Fricative Bookshop	3	3	60
8042	Bookbeat	4	4	80

Delete Queries

A *delete query* removes records from a table. It allows for deletion of records either completely or based on some condition. For example, if we wanted to delete stores from our `store_sales_summary` table that sold less than 80 units, we would do so using the `DELETE FROM` command along with a `WHERE` clause.

```
DELETE FROM publications.store_sales_summary
WHERE Qty < 80;
```

If we select all records from this table again, we can see that the stores with less than 80 units sold have been removed.

```
SELECT *
FROM publications.store_sales_summary;
```

StoreID	Store	Orders	Items	Qty
7066	Barnum's	2	2	125
7067	News & Brews	2	4	90
7131	Doc-U-Mat: Quality Laundry and Books	3	6	130
8042	Bookbeat	4	4	80

If you want to clear a table out completely and still maintain the table structure and the data types for each field, you can just drop the `WHERE` clause from your delete statement.

```
DELETE FROM publications.store_sales_summary;
```

The table structure still exists, but it no longer contains any data.

```
SELECT *
FROM publications.store_sales_summary;
```

StoreID	Store	Orders	Items	Qty

Append Queries

Another useful action query is the *append query*. As the name implies, an append query adds records to a table. For example, if we wanted to repopulate our now empty `store_sales_summary` table, we could do that using the `INSERT INTO` command and the same query we originally used to produce the results.

```
INSERT INTO publications.store_sales_summary
SELECT stores.stor_id AS StoreID, stores.stor_name AS Store, COUNT(DISTINCT(ord_num)) AS Orders,
COUNT(title_id) AS Items, SUM(qty) AS Qty
FROM publications.sales sales
INNER JOIN publications.stores stores ON stores.stor_id = sales.stor_id
GROUP BY StoreID, Store;
```

Now when we select all from this table, we can see that all the records are populated again.

```
SELECT *
FROM publications.store_sales_summary;
```

StoreID	Store	Orders	Items	Qty
6380	Eric the Read Books	2	2	8
7066	Barnum's	2	2	125
7067	News & Brews	2	4	90
7131	Doc-U-Mat: Quality Laundry and Books	3	6	130
7896	Fricative Bookshop	3	3	60
8042	Bookbeat	4	4	80

Update Queries

The last type of query we will cover in this lesson is the *update query*. An update query changes values stored in a table. For example, suppose each store sold 5 more units today and we want to update the quantities in our `store_sales_summary` table. The update query below will add 5 units to the values in each store's Qty column.

```
UPDATE publications.store_sales_summary
SET Qty = Qty + 5
```

If we select all from the table, we can see the updated quantities.

```
SELECT *
FROM publications.store_sales_summary;
```

StoreID	Store	Orders	Items	Qty
6380	Eric the Read Books	2	2	13
7066	Barnum's	2	2	130
7067	News & Brews	2	4	95
7131	Doc-U-Mat: Quality Laundry and Books	3	6	135
7896	Fricative Bookshop	3	3	65
8042	Bookbeat	4	4	85

We can also choose to only update values where certain conditions are met by adding a `WHERE` clause to our query. For example, if we want to add 10 additional units but only to the stores that have sold less than 100 units, we would do so as follows.

```
UPDATE publications.store_sales_summary
SET Qty = Qty + 10
WHERE Qty < 100
```

Once we select all records from the table again, we can see the updated values for just those records.

```
SELECT *
FROM publications.store_sales_summary;
```

StoreID	Store	Orders	Items	Qty
6380	Eric the Read Books	2	2	23
7066	Barnum's	2	2	130

StoreID	Store	Orders	Items	Qty
7067	News & Brews	2	4	105
7131	Doc-U-Mat: Quality Laundry and Books	3	6	135
7896	Fricative Bookshop	3	3	75
8042	Bookbeat	4	4	95

Summary

In this lesson, we went beyond simple select queries and learned how to leverage action queries to perform different operations in our MySQL databases. First, we learned how to create new tables using make table queries. Then we learned how to remove records from tables with delete queries. We also learned how to insert records from a query result into an already-existing table using append queries. Finally, we covered updating values in tables using update queries. Now that you understand and know how to use these intermediate SQL concepts, you should have the tools you need to construct complex queries and string them together into an intuitive analytical process.