databricks Academy

DAWD 01-4 - Demo - Schemas Tables and Views on Databricks SQL

The two fields below are used to customize queries used in this course. Enter your schema (database) name and username, and press "Enter" to populate necessary information in the queries on this page.

Schema Name:	hive_metastore.class_013_odg7_da_dawd
Username:	class+013@databricks.com

Lesson Objective

At the end of this lesson, you will be able to:

- Describe how to use Databricks SQL to create schemas, tables, and views

Create Schema

I want to show you how to create a new schema (database). In your organization, you may not have the proper permissions to do this, and in this course, you don't have the proper permission, but I wanted to show you how to do this just the same.

1. Run the code below.

The code runs three statements. The first drops the schema just in case we are running this twice. The second creates the schema. Note that there is no `CATALOG` provided. Databricks is set up to use a default catalog, and this is set up by your Databricks Administrator.

The third statement runs a `DESCRIBE SCHEMA EXTENDED`, which gives us information about the schema, including the location where managed table data will be stored.

```
DROP SCHEMA IF EXISTS hive_metastore.class_013_odg7_da_dawd_schema CASCADE;
CREATE SCHEMA IF NOT EXISTS hive_metastore.class_013_odg7_da_dawd_schema;
DESCRIBE SCHEMA EXTENDED hive_metastore.class_013_odg7_da_dawd_schema;
```

Copy

Managed Tables

Create Table

From here on out, you have the proper permission level to complete what I'm demonstrating for you here.

Let's create a managed table in our schema and insert some sample data.

Note that I have "`USING DELTA`" at the end of the `CREATE` statment. This is optional because Delta is the default table type.

3. Run the code below.

```
USE hive_metastore.class_013_odg7_da_dawd;
CREATE OR REPLACE TABLE managed_table (width INT, length INT, height INT) USING DELTA;
INSERT INTO managed_table VALUES (3, 2, 1);
SELECT * FROM managed_table;
```

Copy

View Table Information

4. Run the code below to see information about the table we just created.

   Note that the table is a managed table. When we drop this table, our data will be deleted. Note also that this is a Delta table.

```
USE hive_metastore.class_013_odg7_da_dawd;
DESCRIBE EXTENDED managed_table;
```

Copy

DROP the Table

5. Run the code below to drop the table.

```
USE hive_metastore.class_013_odg7_da_dawd;
DROP TABLE IF EXISTS managed_table;
```

Copy

External Tables

There is a dataset, called Sales, that is currently in an object store. Running the command below creates an external table that is associated with this dataset.

5. Run the code below.

   The table's data is stored in the external location, but the table itself is registered in the metastore. We can query the data just like any other table in the schema.

```
USE hive_metastore.class_013_odg7_da_dawd;
CREATE TABLE external_table
    LOCATION 'wasbs://courseware@dbacademy.blob.core.windows.net/data-analysis-with-
databricks/v02/retail-org/sales/sales_delta';
SELECT * FROM external_table;
```

Copy

View Table Information

Let's look at the table's information

6. Run the code below.

Note that the table's type is `EXTERNAL`, and the table's location points to the file system. Dropping this table will not affect the data in this location.

```
USE hive_metastore.class_013_odg7_da_dawd;
DESCRIBE EXTENDED external_table;
```

Copy

Dropping External Tables

The command below will drop the table from the schema.

7. Run the code below to drop the table.

Note that we dropped the table, so we won't be able to query the data using the kind of `SELECT` query you may be used to using.

```
USE hive_metastore.class_013_odg7_da_dawd;
DROP TABLE IF EXISTS external_table;
```

Copy

Dropping External Tables Does Not Delete Data

8. Run the code below.

Even though we dropped the table, we are still able to query the data directly in the filesystem because it still exists in the object store.

Now, this is one of the coolest features of Databricks SQL. We've talked about how the use of schemas and tables is just an organizational contruct. The data files located in this location can be queried directly, even though they are not part of a table or schema. We use tables and schemas simply to organize data in a way familiar to you.

```
SELECT * FROM delta.`wasbs://courseware@dbacademy.blob.core.windows.net/data-analysis-with-
databricks/v02/retail-org/sales/sales_delta`;
```

Copy

Views

Views can be created from other views, tables, or data files. In the code below, we are creating a view from the data in the data file we used above.

9. Run the code below.

The view now gives us all the sales that totaled more than 10,000. If new rows are added to the Sales data file, the view will update every time it's run.

```
USE hive_metastore.class_013_odg7_da_dawd;
CREATE OR REPLACE VIEW high_sales AS
    SELECT * FROM delta.`wasbs://courseware@dbacademy.blob.core.windows.net/data-analysis-with-
databricks/v02/retail-org/sales/sales_delta`
        WHERE total_price > 10000;
SELECT * FROM high_sales;
```

Copy

Common Table Expressions (CTEs)

Here is an example of a CTE. We are going to query the view we just created, but we are going to filter based on the `total_price` being above 20000. We are calling that `sales_below_20000` and then immediately querying the `DISTINCT customer_name` from that CTE.

```
USE hive_metastore.class_013_odg7_da_dawd;
WITH sales_below_20000 AS
    (SELECT *
     FROM high_sales
     WHERE total_price < 20000)
    SELECT DISTINCT customer_name FROM sales_below_20000;
```

Copy